Services, contracts, policies and eCommunities – Relationship to ODP framework

Lea Kutvonen and Janne Metso

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

Abstract— **Agility for inter-enterprise collaborations requires development of interoperability and B2B middleware services. In the Pilarcos and web-Pilarcos projects, such middleware solutions have been researched and developed, in close relationship to the ODP reference model and complementary standards. Although it has been claimed that ODP reference model has not reached its audience, most of the topical trends – service oriented computing (SOA, SOC), inter-enterprise business process management, virtual organizations management, and Web Services – reflect the same foundations.**

This paper discusses the issues around the concepts of services, eCommunities, and contracts as they are visible in the web-Pilarcos architecture. The contribution is directed two ways: for enhancing the concept related to service within the ODP framework, and for showing how the web-Pilarcos architecture applies the distinct concept of service type for gaining improved interoperability control over what is available for example with Web Services.

I. Introduction

The web-Pilarcos architecture provides middleware level services for establishing and controlling inter-enterprise collaborations (virtual organizations) also called eCommunities. The eCommunities involve a set of autonomous enterprise-application level services, and are controlled by eCommunity contracts.

The eCommunity contracts capture metainformation about the community structure (roles and responsibilities of the participants, behaviour in terms of interactions between the roles) in the form of an agreed business network model, agreed policies restricting that model behaviour, and information about the current participants (technical such as access information, and business oriented such as cost of service, trustworthiness of the partner). The eCommunity contract also provides for controlled methods for renegotiating the contract and making changes to partnerships, policies, technical details, etc. The contracts capture information from all ODP reference model viewpoints, ranging from enterprise (and business or legal) concerns to engineering (and technology) aspects.

The web-Pilarcos middleware architecture [1] applies many of the topical patterns for interoperable systems: SOA (service oriented architecture) [2] and Web Services [3] use similar separation of service offers as announcements of available services, and dynamic discovery of partners into compositions of complex services. The architecture is also concerned with operational time interoperability monitoring, and on mechanisms allowing flexible, community-wide resolution of breaches.

This paper brings forth the concepts related to services, service offers, and eCommunity contracts, continuing the discussion [4] of applying ODP concepts to inter-enterprise collaboration. The discussion hopefully enlightens the potential expansions on these concepts in the ODP reference model [5]–[8]. In addition, the required facilities for supporting these concepts are outlined, showing some omissions in Web Services arena. The paper is structured as follows. Section II outlines the web-Pilarcos architecture and services. The essential concepts of service, service offer, and eContract are further discussed in Sections III and IV. As the architecture can be heavily criticized on too high costs, the Section V is included to show some practical measurements on the eCommunity establishment phase.

II. The B2B interoperability middleware

The web-Pilarcos architecture is designed to compose and govern composition of services provided by autonomous enterprises. The architecture is a federated one (in contrast to unified or integrated), assuming that services are developed independently and their interfaces and properties are compared during collaboration establishment and monitored for conformance during the operational time of the collaboration. (Unified model would make an assumption that the services would be produced using a shared collaboration model.) Figure 1 illustrates the setup. The B2B middleware is to provide business-applications with concepts and practical infrastructure-level services for eCommunity management through agents dealing with contracts, and transparently ensuring interoperability using both statical and dynamic techniques. The architecture does not provide a shared execution platform or enactment of the business network models. Instead, the business applications themselves are expected to include a technique forwarding their internal/local workflows as triggered either by peer requests or by internal triggers. A set of metalevel protocols between middleware level agents for the eCommunity management tasks is completely separate from the application protocols.

The issues to be addressed are interoperability between services and management of the lifecycle of the collaboration. The lifecycle management involves

- population of the eCommunity in such a way that the participating services are interoperable as discussed below, and a negotiation cycle between participants to

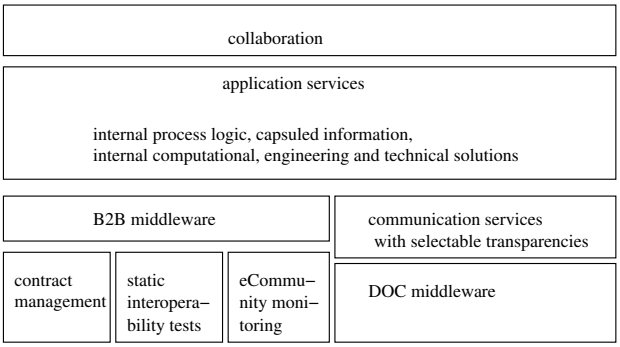

Fig. 1. Architecture overview.

agree or refine the suggested eCommunity properties; the population process selects from a service offer repository a suitable offer for each role in the business network model suggested by the initiator (if the business network model is not appropriate, the process fails);

- establishment of the eCommunity so that all participants are technically prepared for providing services and have done all necessary contract management chores;
- monitoring of potential breaches (non-conformance to the behaviour specified by the contract) in the interactions between participating services;
- monitoring the global, coarse-grain progression of the collaborative work in the eCommunity;
- reacting by renegotiations and contract changes for change requests initiated by involved parties;
- termination of the eCommunity either by the completion of the collaboration goal, by breach resolution, or timely termination of the contract.

In the management of eCommunities, interoperability is a prominent issue [9], [10]. Interoperability, or capability to collaborate, means effective capability of mutual communication of information, proposals and commitments, requests and results. Interoperability covers technical, semantic, and pragmatic interoperability. Technical interoperability means that messages can be transported from one participant to another. Semantic interoperability means that the message content becomes understood in the same way by the senders and the receivers. This may require transformations of information representation or messaging sequences. Finally, the pragmatic interoperability captures the willingness of partners for the actions necessary for the collaboration. The willingness to participate involves both capability of performing a requested action, and policies dictating whether the potential action is preferable for the enterprise to be involved in. In the pragmatic view, process-awareness in terms of collaborative business process model is needed, augmented with nonfunctional aspects, some of which are related to business policies.

At the pragmatic level, issues on the business strategies, values, and rules become visible. Partially these are reflected by using business network models as a founding element in the eCommunity contracts [4]. In addition, policies and properties embedded in to the eCommunity contracts and service offers are used for carrying this kind of information. Technically, most of those features appear as plain name-value pairs, but development of the carried semantics requires large scale ontology and metrics development at international consortia appropriate for each industrial area separately. This theme is further discussed in Section IV.

III. Services and service types

We assume that business-application services are independently implemented and deployed on the enterprises' computing system. The implementations can be generated using MDA style tools, using specifications of collaborations as a starting point. However, existing legacy software can be used equally.

The available implementations establish the practical capabilities for a service in an enterprise. The enterprise can well support multiple, slightly different implementations for the same kind of service. Reasons for this multiplicity can be strategic (different business rules embedded), or technical (different platform and communication facilities used), or evolutionary (old and new software used side by side till the old can be made obsolete as clients have become able to interoperate with the optional ones).

Thus, the *service (implementation)* refers to the computational composition of application software that provides a complete business service, or supports a step in such.

The ODP reference model differentiates between types and templates for objects [5, clauses 9.7 and 9.11]. This separation is heavily used in our approach. The *service templates* are only used within an administrative domain responsible of executing a service. An *administrative domain* is a term related to federations [6, clauses 10.3, 5.1.1, and 5.1.2] and we use it for denoting a technologically consistent system within an enterprise. Outside that administrative domain, the service template has no relevance, but instead, a less restrictive concept of *service type* is of importance.

The ODP reference model indicates two methods by which objects can become known in a system, namely *instantiation* and *introduction* [5, clauses 9.13 and 9.16]. Instantiation is needed within the administrative domain responsible of running a service, but at other domains, the presence of that service is created though introduction. The introduction process is required to reveal the type of object in question, i.e, associate the object with a useful predicate it matches to.

In an inter-enterprise environment, or a SOA environment, the service offer repositories (UDDI [11], ODP trading service / OMG trader [7]) reflect the introduction process. All service offer repositories expect exporters of offers to define the identity of the service, its service type reference, and access information. The service type denotations vary, as well as the level of control in their definition. In the (web-)Pilarcos case, the service offers are expected to include in addition attributes as defined by the service type and the middleware itself (in relation to environment contract, see below).

In the web-Pilarcos architecture service types are stored into a type repository, and only those previously defined type

descriptions can be associated with service offers. The service type descriptions can be published either as part of a design process or independently, by various enterprises in the global network, and have to be verified before acceptance to the repository [12], [13].

Service types are abstract descriptions of business service functionality and they define functional and non-functional properties for a class of business services. Functional part of a service type comprises of an interface signature, an interface protocol which describes the service behaviour and additionally semantic annotations for exchanged documents (messages). Engineering level information, such as binding of a service instance into a specific communication protocol or address, is not part of the service type. The non-functional properties of a service type describe issues on business level concerns, QoS requirements, and policies. The non-functional aspects are given as service attributes (name-value pairs), where each element denotes a semantic concept from a shared ontology.

Technically, the service type defines

- the service interface signatures;
- associated to the interfaces, restrictions on the ordering of invocations (a loose way of expressing a protocol, only giving restrictions on the necessary ordering relationships)
- associated with the interfaces, information contents of the messages; and
- set of attributes associated to the service, labeled either optional or mandatory.

Thus the service type repository is actually defining an application-area specific, service-type based ontology for information about the business and legal aspects of the service. The attributes can reflect the cost of service, availability, expectations on the business networks involved, and legislation to be used for evaluating the service and managing breaches. As some of the attributes are optional, this gives flexibility in making service offers with different kind of business networks in mind. Different collaborations will depend on different attributes.

For example, Web Services technologies are still missing type repository type of services. The benefit that is provided lies in the trustworthiness of the type definition. For a repository, some quality requirements can be placed: static verification of the models, persistent and continuous availability of items, correctness of assertions on relationships between types, etc.

Besides the attributes, the service offers are expected to contain attributes that form an offer of environment contract [12]. The *environment contract* [5, clauses 11.2.3 and 13.2] of ODP refers to commitments between an object and its environment, i.e., prerequisites under which the service in question can be provided. In the web-Pilarcos case, the environment contract part of the service offers include selection of binding type, channel type, and configuration parameters for the abstract communication layer depicted in Figure 1. The binding type defines what services the abstract communication layer is expected to provide and what role-related interfaces for communicating peers are provided. The channel type denotes the architecture of the communication channel so that the middleware is able to configure it appropriately. In addition, requirements on shared computing or information repository resources can be placed.

IV. eCommunities and business services

In the eCommunity establishment process, the business level commitment to the contract takes place, as well as the computational setup of the collaboration. There are two sources for regulating information involved: the business network model, and the service offers selected for the contract.

The business network model we use is a expansion of the *ODP enterprise description* [14], [15]. Several, functionally separate *community descriptions* are integrated by denoting which of the roles have to overlap for the network. Each functional community is defined in terms of roles comprising of a role name, service type required, and free form constraints on a) properties of the service offer to be selected to that role, and b) rules to be used for monitoring the service behaviour during the eCommunity operation. In the population process, the service type requirement and selection criteria are used for retrieving service offers. The environment contract and policy requirements in the service offers further cause interdependent requirements, thus making the matching process quite complicated. Although this process is at first sight very expensive, it stays within reason, due to the time-based and memory-space restrictions on the searches. For further evidence, Section V shows measurements done on top of a CCM platform.

At the level of eCommunity, the relationship between the business environment and the provided service becomes another aspect of the environment contract, thus forming another defining aspect of the service itself. As the participants in the eCommunity are in some extent dependent on each others' services, the commitments they make are made on the premises that others fulfill their commitments according to the contract too. Therefore, the concept of business service cannot be fully defined in isolation, but is dependent on the business network it is part of.

A *business view of the service* can be seen as a computational service, associated with environmental constraints from the providing enterprise's computing environment, its role in the business network in question, and the business and computing restrictions set by its peers in the business network.

Because business network models gain such an important role in interpretation of all service related concepts, the web-Pilarcos architecture provides a repository for these models. The benefits of the globally available set of verified models (with related service types in the type repository) arise from the facilities for reasoning from the process models, process-level interoperability support in the middleware, and guidance to the service markets.

Having captured all these aspects to the eCommunity contract, we still cannot claim that the correct business behaviour or legally correct interoperation is guaranteed. The models

provided for interoperability checking are just models indicating the intention of the service provider. Even if the models were used to generate the software elements from those specifications, there would be uncertainties involved. Therefore, monitoring of the behaviour conformance against the model is essential. Depending on the level of distrust and acceptable performance penalties, the conformance requirements need to be selected appropriately. The monitoring aspects are presented in [16], so they are not discussed here further.

Mechanisms to enforce conformance to business needs and interoperable computing requirements must involve dynamic methods in addition to the traditional static analysis. One of the essential reasons for this is the need for capturing enterprise policies and business rules. Those can change any time, disregard of the eCommunity contracts in use – thus causing contradictions at operational time. Furthermore, the ontologies for eCommunity contracts may omit some of the locally important policies, and therefore be unaware of the contradiction.

We use the following terms. Enterprise policies are used to govern the resources and services of an enterprise. Enterprise policies may oblige, permit, or prohibit access to some service or information. Enterprise policies or eCommunity policies are used to modify the basic business process models; negotiation of a policy value is used to refine which alternative path through the model is in use. Business rules are declarative statements that define or constraint some aspect of business service behaviour. Business rules may define different pricing policies or service availability for different client categories. A business rule typically affects the non-functional aspects or information exchanged in a service interaction.

The above definition of business service did not explicitly mention the policies. However, policies are an integral part of the context of the business service, and is to be seen as part of the environmental constraints. Therefore, policies come as a time dependent and enterprise specific element into the picture.

Finally, a few words on another time dependent property. One of the very essential properties of a business service is that of trust. In the TUBE project [17], we are extending the web-Pilarcos architecture with trust concepts. Decisions on trusting a business service is made first at the establishment of an eCommunity; business service trustworthiness is estimated by reputation information and local experience of that enterprise and that business service especially, and has to exceed a treshold dependent on assumed risks and importance of the eCommunity participation for that enterprise. As the correct behaviour of the peers in the eCommunity are not completely trusted even after this, the behaviour is monitored, and new experience information and new reputation information is generated by successes and breaches of conformance.

V. Cost of eCommunity establishment

While Pilarcos middleware provides facilities for dynamically forming eCommunities there is overhead cost from the use of the middleware services. This cost has been estimated by a series of performance measurements on the prototype application and infrastructure services. First, the overall overhead of Pilarcos middleware in terms of CPU load and effect on response times seen by a client was considered. Second, the main elements of the cost were factored out at the areas of business network population, eCommunity establishment and termination and adaptation of communication across the CCM and EJB platforms. Third, some scalability tests were run to see how the input rate of client requests affect the response times. Because the Pilarcos middleware feasibility is much dependent on the scalability of Pilarcos enhanced trading, a set of separate measurements were done on the population process alone.

The measurements were performed with a system that included all the Pilarcos infrastructure services and an application case. The application case does not include much of application-specific processing, but is there mainly to generate a meaningful mixture of requests to the middleware under study.

In concrete terms, the Pilarcos prototype consists of Pilarcos infrastructure service components and Tourist Information Service application components that take advantage of the Pilarcos services. The infrastructure services and most of the application components have been implemented on two new CORBA Component Model platforms, the Java-based OpenCCM [18] and the C++-based MicoCCM [19]. In addition, one of the application domains is built on Enterprise JavaBeans technology, running on the JBoss [20] application server.

The sample case study is built around an idea of a portal service, the *Tourist Info*, which provides traveller access to vertical tourist services like travel information, hotel bookings, and weather services. It is assumed that neither the portal service nor the vertical services are free, and the traveller has some electronic payment instrument (e.g. credit card) available. Figure 2 shows the business entities and the communities formed by them. The example involves two business communities: a tourist info community and hotel info community. The tourist info community contains three domains each with their own role (tourist info client, tourist info service, payment service) and describes the business community related to providing the portal service. The hotel info community also contains three domains each with their own role (hotel info client, hotel info service, payment service) and describes the business community related to providing the only implemented vertical service (hotel bookings). The tourist info client and the hotel info client represent the users of corresponding services whereas the tourist info service and the hotel info service represent the providers of the services. Payment service represents a trusted third party used to mediate the transfer of funds between the other entities in the community.

The measurement environment consists of seven 1GHz Pentium III workstations with 512MB RAM memory. Workstations were connected with closed 100 Mbps Ethernet. The operating system used in the workstations was CS Linux with kernel 2.4.18. Different components of the Pilarcos

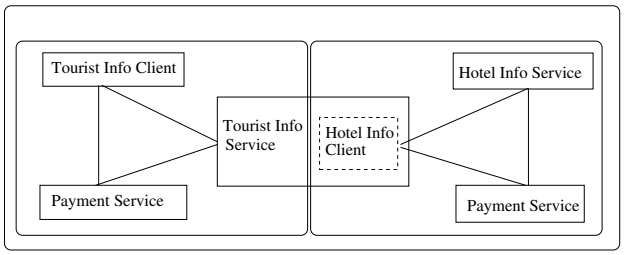

Fig. 2. Business entities and communities in the *Tourist Info Service* case [21].

prototype were distributed to the workstations. The components related to one domain were deployed in one machine with the exception that in Hotel service -domain CCM and EJB components were distributed to two separate machines. Background clients used two additional machines and were evenly distributed in them. Before measurements the system was warmed up with 3000 measurement rounds done by the client. The measurements themselves included 8000 rounds. Each benchmark used one measurement client and two to six background clients. Throughput optimization flags were used for Java virtual machines. In all measurements the IBM Java 1.4.1-platform was used. For the Pilarcos trader measurements, the client program was run on one workstation and the Pilarcos trader on another. Java-based ORBacus trader 2.0.0 [22] was running in compiled mode on the same workstation as the Pilarcos trader.

Looking at the measurement results, the cost of Pilarcos middleware use appears to be acceptable. As we consider a "round" from a user, the application runs the following steps:

- ask the Pilarcos trader to populate a business network;
- tell the business network manager to create the community; this actually involves the population and creation of the second community in the application scenario;
- run a group of application scenario-specific operations and requests for the servers;
- end the session and ask the business network manager to terminate the communities.

The population, eCommunity establishment, and eCommunity termination phases can be seen in Figure 3. Under normal load conditions, the population process takes less than 25 milliseconds, eCommunity creation under 40 milliseconds (the measured 100 ms includes two eCommunity creation operations and a population operation), and eCommunity termination about 20 milliseconds (again, the measured numbers cover two terminate operations). Time consumption for getInformation and makeReservation operations are also noticeable in the measurements. This is caused by the fact that the execution process of these operations includes several service calls between components residing in different domains.

For the measurements presented in Figure 3 the number of request-making client computers were increased so that the input rate of requests reached a level where saturation on one of the computers was approaching. However, no unexpected

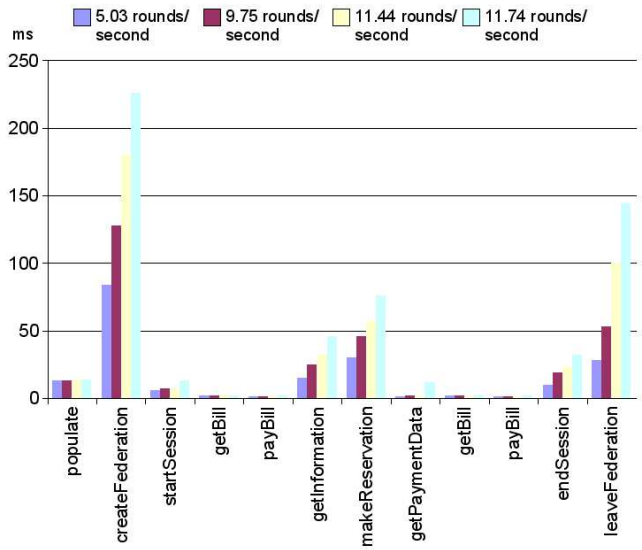

Fig. 3. Response times for the client [21].

behaviour on response times is seen here.

Figure 4 show that the Hotel Server domain is becoming a bottleneck in the system as the load increases. Within the Hotel Server domain the CCM platform running the infrastructure services is a bit more loaded than the EJB platform running the application components. This ratio depends, however, on the computational complexity of the applications.

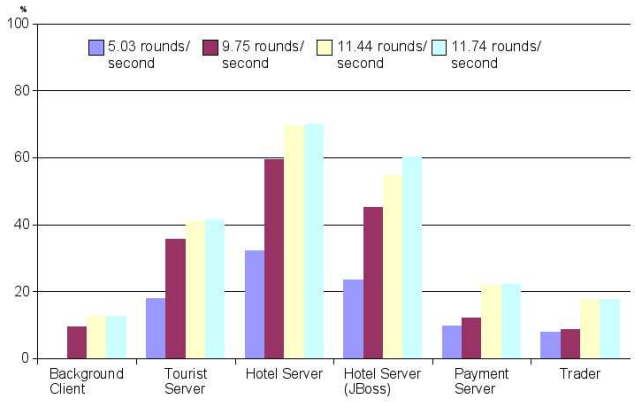

Fig. 4. Processor load in servers [21].

The conclusions from this behaviour are twofold. First, additional measurements are needed in a more realistic and complex environment where the network latencies are longer (e.g. Internet). Second, in the current implementation the cost of adapter creation is considerably high for interactively used applications. A library-based solution has been planned but was not yet available for measurements.

To support interpretation of the above measurement results, Figure 5 shows the same application components running without the help of Pilarcos middleware. The configuration of

application components is here fixed, and only OpenCCM is used as a platform. Thus, the flexibility of finding appropriate partners for the eCommunity is missed, and also heterogeneity support. All application components have been selected beforehand and their locations are resolved by a name server at the start-up time. The measurements show that the overhead load is mostly visible in the new operations offered by the Pilarcos middleware. An additional few milliseconds fixed cost is visible on all operations.

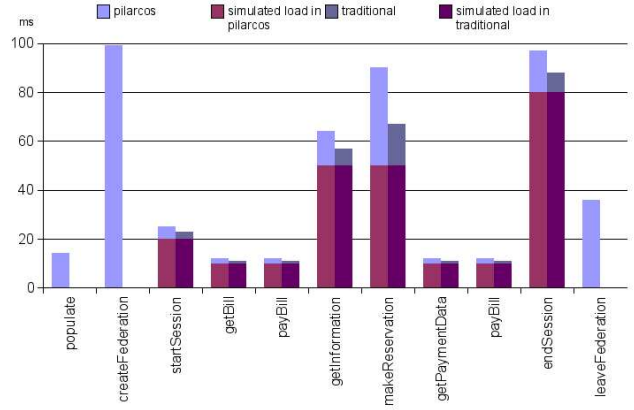

Fig. 5. Time used in different phases seen by client with simulated load [21].

In general, the measurement on the full prototype software provided results much like we expected. However, platforms are not mature enough and caused scalability problems themselves, so scalability tests were not done as thoroughly as we wished.

More thorough measurements could be done on the Pilarcos trader. In this paper we only bring up a few comments, and refer to [21], [23] for details.

The population process was able to provide one eCommunity proposal in an average of 22 milliseconds. In addition, transferring the request and the result over CORBA took an average of 30 ms, raising the total to 52 ms. Most of the transferring time was result of marshalling delays.

These times were not much affected by the number of service offers in trader database. With offers distributed evenly between the roles, no significant effect on the search time was seen with database sizes of up to 2550 offers. This behaviour was expected, since the population algorithm never needs to search the entire offer database. Instead, the search time is directly proportional to the number of requested eCommunity proposals.

Search times grow in linear proportion to the number of policies in service offers. In the baseline case there were 32 policies per offer, which is rather a high estimate.

The cost of marshalling is fairly high, because the eCommunity proposal data structure is designed for readability and flexibility, not speed. It contains several nested sequences and makes heavy use of the CORBA `Any` type, making marshalling very resource consuming. The marshalling delay could be significantly reduced by using known CORBA IDL optimization techniques. For the rest of the results, only the time spent in the search process is presented, since the marshalling delay is constant and predictable.

One of the potential problems were cases where there are very few acceptable service offers in the trader database. However, we found that both the average response time and its variation grow significantly with low match ratios, but are still tolerable even at a match ratio of 5 %. Based on these results, the practical usage area for the Pilarcos trader is with offer match ratios at and above 5 %. Figure 6 shows this effect.

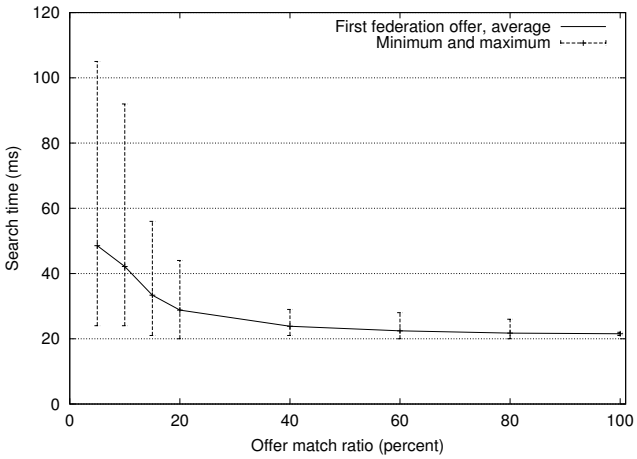

Fig. 6. Effect of offer match ratio on search time.

Figure 7 illustrates the effect of varying the number of roles in the business network model. Again, the dependency is linear, as expected. Based on these results, large network models with up to ten roles would be practical; most probably, for such large networks, other aspects than the population process are more significant.

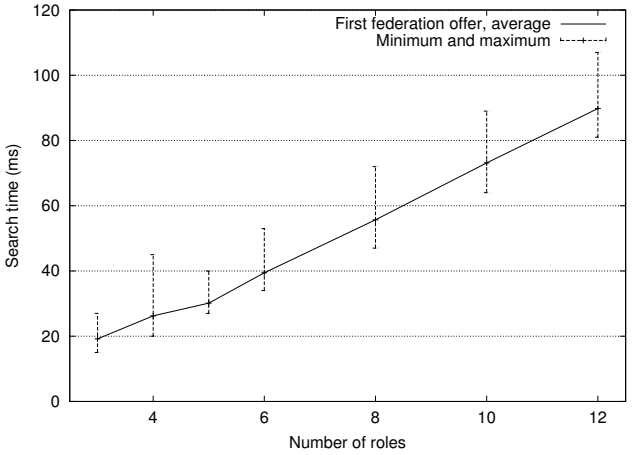

Fig. 7. Effect of number of roles on search time.

The above results were based on measurements on a standalone Pilarcos trader that includes its own offer database. We also tried the effect of using ORBacus trader for storing the

service offers. The search times are nearly ten-fold compared to the standalone case, with significantly larger variation. These are results from the block-wise transfer of service offers from the ORBacus trader. Transferring service offers between the ORBacus trader and the Pilarcos trader takes more than half of the ORBacus trader query time. When a CORBA trader is used as a back-end, a significant speedup could be achieved by collocating the Pilarcos trader and the CORBA trader in the same process. This trial is interesting because it shows an effect that is expected when using federated (linked) CORBA traders.

VI. Conclusion

This paper discusses concepts of computational and business services, and illustrates how business network models and service types bring in an ontological route for enforcing appropriate properties to be brought in to the eCommunity contracts. The properties spanning over all five ODP viewpoints capture regulations and restrictions as well from business strategical as from computing system perspective and bring them to the operational environment to reflect the state of the eCommunity.

Infrastructure services like global service type repositories and business network model repositories, and trust management facilities are the latest contributions to the arena of inter-enterprise interoperability and federated architectures.

A set of practical measurements on the prototype infrastructure services show, that the federated architecture model is not infeasible in practice, and also illustrates that the basic concepts for eCommunity management can be implemented on current distributed computing platforms.

Acknowledgment

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 work much integrates with RM-ODP standards work, and recently has found an interesting context in INTEROP NoE collaboration.

References

[1] L. Kutvonen, T. Ruokolainen, J. Metso, and J. Haataja, "Interoperability middleware for federated enterprise applications in web-Pilarcos," in *INTEROP-ESA'05*, 2005.

[2] M. P. Papazoglou and D. Georgakopoulos, "Service oriented computing," Oct. 2003.

[3] B. Benatallah, O. Perrin, F. Rabhi, and C. Godart, *Web Service Computing: Overview and Directions*. Springer Verlag, 2004, ch. xx.

[4] L. Kutvonen, "Challenges for ODP-based infrastructure for managing dynamic B2B networks," in *Workshop on ODP for Enterprise Computing (WODPEC 2004)*, A. Vallecillo, P. Linington, and B. Wood, Eds., 2004, pp. 57–64. [Online]. Available: http://www.lcc.uma.es/~av/wodpec2004/WODPEC2004-Proceedings.pdf

[5] *Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing. Reference Model of Open Distributed Processing. Part 2: Foundations*, ISO/IEC JTC1, 1996, iS10746-2.

[6] *Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing. Reference Model of Open Distributed Processing. Part 3: Architecture*, ISO/IEC JTC1, 1996, iS10746-3.

[7] *IS13235 ODP Trading function.*, ISO/IEC JTC1, 1997.

[8] *Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing. Reference Model of Open Distributed Processing. ODP Type repository function*, ISO/IEC JTC1, 1999, iS14746.

[9] L. Kutvonen, "Automated management of interorganisational applciations," in *EDOC2002*, 2002.

[10] T. Ruokolainen and L. Kutvonen, "Interoperability in service-based communities," in *First International Workshop on Enterprise and Networked Enterprises Interoperability (ENEI2005), in conjuction with BPM2005*, 2005, to appear.

[11] OASIS Consortium, *Universal Description, Discovery and Integration of Web Services (UDDI) 3*, 2002, http://www.oasis-open.org/committees/uddi-spec/tcspecs.shtmluddiv3.

[12] L. Kutvonen, "Trading services in open distributed environments," Ph.D. dissertation, Department of Computer Science, University of Helsinki, 1998.

[13] ——, "Relaxed Service-type matching and Transformation management," in *Workshop on Enterprise Modelling and Ontologies for Interoperability, EMOI-INTEROP 2005*, June 2005, poster paper accepted for publication.

[14] *IS15414 ODP Enterprise Language*, ISO/IEC JTC1, 2003.

[15] *Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing. ODP Enterprise Language, Ammendments*, ISO/IEC JTC1, 2003, pAM13235.

[16] J. Metso and L. Kutvonen, "Managing Virtual Organizations with Contracts," in *COALA2005 workshop*, June 2005, submitted manuscript.

[17] L. Viljanen, S. Ruohomaa, and L. Kutvonen, "The TuBE approach to trust management," in *Proceedings of the 3rd iTrust internal workshop*, 2004, to appear.

[18] "OpenCCM web-site (LIFL)," Dec. 2001, http://www.lifl.fr/OpenCCM.

[19] "MicoCCM web-site," Dec. 2001, http://www.fpx.de/MicoCCM.

[20] "JBoss web site," Jan. 2003, http://www.jboss.org.

[21] M. Vahaaho, J.-P. Haataja, J. Metso, T. Suoranta, E. Silfver, and L. Kutvonen, "Pilarcos prototype II," Department of Computer Science, University of Helsinki, Tech. Rep. C-2003-12, Jan. 2003.

[22] *ORBacus Trader, version 2.0.0*, IONA Technologies, 2001, http://www.iona.com/products/orbacus/trader.htm.

[23] M. Vahaaho, "Trading with architecture models (Arkkitehtuurikuvauksia hydyntv meklaus)," Master's thesis, Department of Computer Science, University of Helsinki, Feb. 2003, in Finnish.