Interoperability middleware for federated enterprise applications in web-Pilarcos

Lea Kutvonen, Toni Ruokolainen, Janne Metso, Juha-Pekka Haataja

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

Abstract. Electronic business networks have become necessary for the success of enterprises. Architectures and tools for lowering the cost of collaboration establishment and improving the facilities for managing and maintaining the networks are under active development. The web-Pilarcos B2B middleware is designed for interoperability of autonomous enterprise applications in interorganisational context. The approach is a federated one: All business applications are developed independently, and the B2B middleware services are used for ensuring that technical, semantic, and pragmatic interoperability is maintained in the business network. In the design, attention has been given on the dynamicity and evolution aspect of the network. This paper discusses the concepts provided for application and business network owners and configurators, and the middleware knowledge for interoperability support.

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 timely to the requirements rising from the changing co-operation networks and their communications needs.

From the computing infrastructure side, the enterprise needs can be addressed by an architecture where business level services, B2B middleware, and abstract communication services are clearly separated from each other, and the relationships between collaboration lifecycle, B2B middleware, and software engineering tools are changed from the traditional approach. By B2B middleware we mean general infrastructure services that provides concepts and operations for forming electronic business networks, eCommunities, and managing their lifecycle.

The B2B middleware concepts and operations should be such, that strategical, process-related and technological needs of electronic business network management is filled. Such needs we believe to include the following

- form new business networks that provide added value services for clients;
- join to multiple networks at the same time without unnecessary restrictions on technologies or operational policies;
- take up new business processes and services cheaply;
- move existing business networks to new phases of lifecycle so that new collaboration forms can be used;
- monitor the progress and correctness of the collaborative processes;
- automate some collaboration establishment and correction events; and

– protect local services and computing solutions from the changes and failures of the collaboration partner services and solutions.

Traditionally, inter-enterprise collaboration has required integration of enterprise computing systems or applications. The business networks are expensive to establish and maintain, and the cost of changes in the network structure or membership is high. The topical integration techniques vary from new generation ERP systems, process-orientation to distributed workflow management systems.

At present, significant amount of research is focusing on virtual enterprise approaches. Virtual enterprises are joint ventures of independent enterprises joining a shared collaboration process. In many projects, like PRODNET [1], MASSYVE [2], FETISH-ETF [3] and WISE [4, 5], the support environment consists of a breeding environment and operational environment. The breeding environment provides facilities for negotiating and modeling the collaboration processes; the operational environment controls the enactment of the processes. Many of the virtual enterprise support environments use a unified architecture approach: there is a shared abstract model to which all enterprises have to adapt their local services.

In contrast to this, the approach in the web-Pilarcos project is federated: enterprises seek out partners that have services with which they are able to interoperate (within the strategically acceptable limits). A collaboration model (business network model, BNM) is used for explicitly expressing what kind of collaboration is wanted and for verifying that each partner has a conformant view on the rules of the collaboration. In addition to the normal business processes required for the collaboration, the agreement needs to capture potential breach situations and recovery processes from them.

In web-Pilarcos, BNMs (business network models) are not used for enactment, but as semantic interoperability verification tool. Enactment of services and local business processes are required features of the service management facilities of each computing system involved in the eCommunity. Service requests or series of tasks can be initiated and performed either according to the control of an application or a local workflow management system. Instead of depriving the applications the possibilities of requesting services from a remote, collaborating enterprise, the applications are built to include these activities. Local policies (obligations, permissions, prohibitions) are then used for stopping unacceptable requests at operational time; this is necessary, as the context (strategic context from the enterprise point of view) can continuously change.

This design choice has been made in order to make the evolution of BNMs and business networks themselves more flexible. Changes in the model to follow indeed require that the model is explicitly available at the operational time, and that there is a synchronization and negotiation mechanism for partners to reach a safe point where new rules can be adopted.

Section 2 discusses interoperabilty challenges in the context of eCommunity management, and Section 3 briefly describes the web-Pilarcos B2B middleware services and repositories. Section 4 addresses the information repositories presented by the web-Pilarcos middleware. Section 5 discusses methods of finding interoperability problems and potential reactions on them. The paper is concluded by commentary on the current prototype status.

2 eCommunity management and interoperability

The web-Pilarcos architecture proposes a model of inter-enterprise collaborations as eCommunities comprising of independently developed business applications. The applications themselves present local business services and processes, and are able to collaborate with other enterprises within those limits.

The strategical requirements of an business network member towards the collaboration are expressed as a metalevel model that defines a set of external business processes. The structure is defined in terms of roles and interactions between the roles. For each role, assignment rules define additional requirements for the service offer that can be accepted to fulfill it, and conformance rules determine limits for acceptable behaviour during the eCommunity operation. The explicit use of such model allows comparison and matching of strategical, pragmatical goals of members in the network.

Our aim is that interoperability is a functionality provided by the middleware services, a transparent aspect for application services. The applications themselves need only to concentrate on the local business logic, implemented on their local computing platform. Collaboration and eCommunity membership aspects together with pragmatic process-awareness, however, require application level concepts and services. The inter-enterprise collaboration management concepts supported by the web-Pilarcos architecture include those of

- an eCommunity that represents a specific collaboration, its operation, agreements and state; the eCommunities carry identities and are managed according to their eCommunity contract information;
- services that are provided by enterprises, used as members in eCommunities, and are made publicly available by exporting service offers; and
- a set of B2B middleware services for establishing, modifying, monitoring, and terminating eCommunities, or looking from the application service point of view, operations for joining and leaving an eCommunity either voluntarily or by community decision.

To support interoperability transparencies the middleware needs to provide a set of repositories forming an evolving knowledge base about metamodels for communities, ontologies of service types, and services. With this info it is possible to match information representation styles, interfaces, and services.

In the control of the eCommunity lifecycle, the essential element is eCommunity contract. The contract comprises of the business network model (to define the network structure), information about the member services at each role, some overview state information about the progress of the external business processes, and methods for changing the contract itself.

The eCommunity contract captures shared metainformation about the collaboration; reflective methods are used to keep the real system at each involved computing site correspondent with the metainformation. At each administrative computing domain, there is a local management agent that holds and uses knowledge about locally deployed services and their various management methods. The local management interfaces are homogenized by a "standard" protocol for requesting the system to prepare for running a service (resourcing), querying about communication points, releasing the service, etc. Likewise, all relevant changes in the real system are notified and thus change the metainformation accordingly. The eCommunity contract is an active object itself, and includes logic that may react to changes in the metainformation and request local sites for further negotiations or changes in the system state.

Establishing a eCommunity, or entering a new service into an existing one, involves interoperability checking. As we wish to prepare for changes and evolution, we cannot expect identical solutions from the peers here, but need to allow communication when intents fall into same "equivalence class" within which transformations can be done. problems that need automated solutions deal with what kind of interceptors are needed, where in the communication channel, and at what cost.

Monitoring interoperability during the operation of the eCommunity requires sensors at each communication channel end. We assume an abstract communication infrastructure with selectable transparencies and support for non-functional aspects. The communication infrastructure is considered as an explicit, mediating service that needs explicitly to be "parametrized" for deciding its internal structure and level of service to be provided. In that frame, each communication endpoint is equipped with a guard. From the service specifications it is known what traffic should be seen and in which order; in principle the rules can be extended to view the acceptability of contents structures and making trust related decisions. The monitoring system reports from detected situations (task started, completed, unacceptable traffic or lack of expected traffic). In monitoring, the challenges lie in the performance of the communication system, the design of monitoring rules, and decision engine.

3 The web-Pilarcos B2B middleware architecture

The B2B-middleware platform provides a) advanced service discovery based on improved services typing and constraint based selection, b) contract based management of collaboration between autonomous services, and c) proactive local monitoring of contract conformance. Furthermore, repositories with relationship to collaboration modeling, software engineering, and deployment present the knowledge base required for B2B interoperability support.

The service elements of the web-Pilarcos middleware architecture address the need of joining four important processes: a) introduction of BNMs to the model repository, and introduction of supporting service types to the type repository; b) software engineering processes to provide implementations that correspond to the known service types and thus are applicable for the known BNMs; c) deployment of services and export of corresponding service offers to traders, effectively making a commitment to keep the service consistent with the service offer; d) eCommunity establishment process using the provided information.

These processes are only loosely interleaved. Business network models and the actual application services can be developed independently from each other; indeed their development form a quite separate profession. In the platform, these concepts have to meet at the service description level.

The B2B middleware elements are illustrated in Fig. 1. The BNM design process involves introduction and verification of new models to be stored into the repositories. Implementation of new services or introduction of legacy applications involves interaction with the type repository. Deployment processes are naturally augmented with service offer exports. These processes feed in metalevel knowledge of potential participants in communities to be formed. The feeding processes are independent from each other, even withdrawing or deprecating information may take place.

The functional elements presented in Fig. 1 address the eCommunity lifecycle management operations. The *Populator* uses a given BNM for ensuring the pragmatic interoperability of partners to a eCommunity; it also uses a set of compulsory aspects in service offers to determine service types, communication channel

requirements, and nonfunctional aspects to be agreed on for the eCommunity. The populator represents a breeding process where services are selected for eCommunity roles. The population process is a constraint satisfaction challenge between candidates' attribute value spaces and constraints given for roles in the business network model. The service type definitions dictate the attributes and attribute value sets necessary to describe the service, and the actual values for each published service is found in service offer repository. As there is dependencies between selected offers in interacting roles (on channels and NFA), the process is complex.

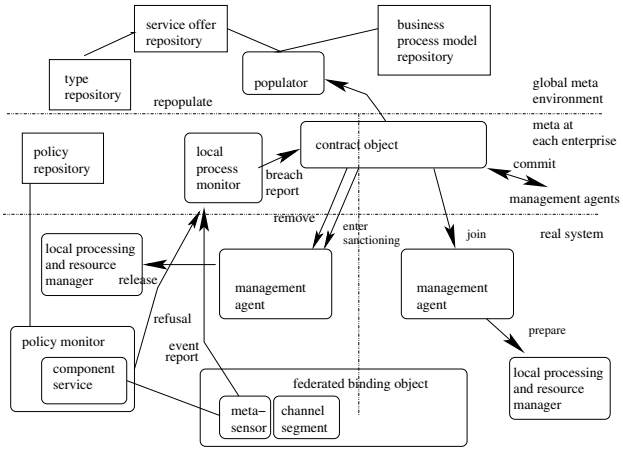

Fig. 1. The web-Pilarcos B2B middleware architecture, and flow of messages when a participant refuses a service due to policy conflict.

The populator provides its clients with a set of interoperable communities, and the suggestions need further to be negotiated on. The populator functionality can be used not only for forming a new eCommunity by discovery of potential partners. Reconfiguration of an existing eCommunity in need of replacement partners, or one partner changing to a significantly different service implementation are also situations where interoperability preconditions need to be checked.

The eCommunity management is done in cooperation with *Business Network Management Agent (BNMA)* and the *Contract* object. The agents are responsible for managing the inter-organizational coordination and management protocols. The contract object is responsible for making decisions regarding the eCommunity is represents. At each administrative domain, there is a BNMA agent to act as a representative between the eCommunity and the local service-providing system. For local administrators the agents provide an management interface for communities. Between themselves the agents have a protocol and interfaces for notifications of task completions, notifications of contract breaches, and negotiation and commitment protocols for joint contract changes. Each local agent receives notifications of contract breaches and task completions from local monitors and propagates this information forward to other agents as needed. Local agent also feeds monitoring instructions to the monitor.

The *eCommunity contract* itself is a key element in the architecture, because it makes available at operational time aspects from different levels/viewpoints of the business network. As can be viewed from Fig. 2 the contract schema captures

as well technical, semantical, (external business) process-related, and pragmatical aspects. Technical information includes service types and related behaviour descriptions, binding types between services, implementation specific messages or function parameters, and policies used in the eCommunity. The structuring element of the contract is the BNM used for the eCommunity: Each role is supplemented with information from participants service offer, each binding with connector parametrization information. Semantical aspects cover information representation formats in messages exchanged. The pragmatic aspects covered include functional description of business processes, policies constraining roles, and non-functional aspects. The non-functional aspects govern features like trust, security, QoS that are traditionally considered as additional platform level service solutions required. In addition, non-functional aspects related to business process models capture more business oriented features, like business rules (captured as policies and monitoring rules here).

```
<complexType name="ContractContent">
  <sequence>
    <element name="architecturePolicies" type="apachesoap:Map"/>
    <element name="bindings" type="apachesoap:Map"/>
    <element name="businessNetworkModel" type="xsd:string"/>
    <element name="contractID" type="xsd:string"/>
    <element name="conversationRecoveryProcess" type="apachesoap:Map"/>
    <element name="description" type="xsd:string"/>
    <element name="endDate" type="xsd:dateTime"/>
    <element name="globalRecoveryProcesses" type="impl:ArrayOf_Process"/>
    <element name="modelPolicies" type="impl:ArrayOf_Policy"/>
    <element name="myRole" type="xsd:string"/>
    <element name="participants" type="impl:ArrayOf_ParticipantInfo"/>
    <element name="rolePolicies" type="apachesoap:Map"/>
    <element name="roleRecoveryProcesses" type="apachesoap:Map"/>
    <element name="startDate" type="xsd:dateTime"/>
    <element name="state" type="xsd:int"/>
    <element name="messages" type="impl:ArrayOf_xsd_anyType"/>
    <element name="sessions" type="impl:ArrayOf_ContractSession"/>
    <element name="offers" type="impl:ArrayOf_ServiceOffer"/>
     ...
  </sequence>
</complexType>
<complexType name="ParticipantInfo">
  <sequence>
    ...
    <element name="coordinator" type="xsd:boolean"/>
    <element name="location" type="xsd:string"/>
    <element name="managementUrl" type="xsd:string"/>
    <element name="name" type="xsd:string"/>
    <element name="role" type="xsd:string"/>            ...
  </sequence>
</complexType>
```

Fig. 2. XML-Schema of contract.

Monitors are part of the communication channel between participating services. A monitor has a generic sensor element that can be configured to filter traffic by classifying it to expected and unexpected event sequences. The BNMA agents provide each monitor a behaviour automata to follow, based on the service choreographies described for the corresponding role. Monitoring reports can be acted on in various ways, scaling from post-operational auditing to proactive prevention of unwanted events. In web-Pilarcos, the intent is to allow major breaches on agreed behaviour or policies to be acted on during the eCommunity operation, and allowing automatic recovery processes to be started. In this respect, the web-Pilarcos approach differs from related projects (like [6]) that otherwise use similar techniques. Because the definition of "severe breach" and the appropriate methods of potentially replacing misbehaving partners are specific to application domain, those rules and

process definitions are compulsory parts of BNMs. These definitions are associated to service, role and BNM level, as shown in Fig. 2.

4 Interoperability knowledge in the global middleware

The three metainformation repositories in the B2B middleware have a central role in establishing an increasing knowledge base that allows interoperability tests on to be made. Each repository is necessarily distributed for scalability and improved accessibility. Due to different type of load, the good distribution styles differ [7].

Ontological support to be incorporated within each repository, and in addition, multiple acceptable relations between concepts in different repositories for reusability of models. It is especially important that service types and BNMs can have separate lifecycles; this provides isolation layers that keep local changes from involving the whole eCommunity and minimizes the effects of BNM enhancements to local services. Furthermore, each model requires only a reasonably narrow expertise to create. In addition to direct relationships between models, the repositories store transformation rules and components for improved transformer/interceptor reusability [7].

4.1 Business network model repository

The BNM repository provides interfaces for publishing new models, verifying their properties, comparing models, and querying models for the basis of population or software engineering processes.

The structure (topology) and properties of a business network are defined by its BNM that explicates the roles of partners and the interactions between roles that are needed for reaching the objective of the eCommunity. A BNM comprises a collection of roles, a set of connectors and a set of architecture specific non-functional properties. The approach combines ideas from ODP enteprise viewpoint language [8] and those of separating functional units and their interconnection into distinct concepts of components and connectors [9].

A role represents a logical business service or entity in an administrative domain. The role definition expresses the functional and non-functional properties required. Role functionality is described as a composition of service types and role specific synchronization patterns. Service types are formal definitions of service interface syntax, communication semantics and behaviour, retrievable from the type repository. Synchronization patterns express causal relationships between actions in distinct services of a role (by setting preconditions for interactions using terms before, after etc).

Interaction relationships between roles of are described by `connections`, each of which include a set of `connector` elements. Each `connector` element defines a bilateral connection between service interfaces. Connector may also describe rules concerning eCommunity coordination. These synchronization patterns are similar to those of e.g. Join-calculus [10]. A synchronization pattern is a rule with a premise and associated activity. For example, a `connector0` between *Seller* and *Delivery* can define a `guard` element, which is used to coordinate the eCommunity in such a way that the *Buyer* is notified whenever a *confirmDelivery* message is received by *Seller*. Connectors may define other communication related properties, such as control or data adaption, eCommunity coordination and non-functional properties of communication.

Nonfunctional properties are managed as named values that ar used for selecting the right technical configurations from the underlying platform. Some properties are used for dynamic brancing of behaviour at operational time. The roots of these decisions are at the business level, but the negotiation and commitment protocols needed are preferrably transparent to the business services.

The web-Pilarcos tools do not provide a modeling tool. Tools designed for business process or workflow modeling could be used to provide the models. However, the usually used structuing of models into processes or flows has to be transformed into a role-focused view. This is necessary as each role will be technically be governed by an autonomous administrative domain.

4.2 Type and service offer repositories

The *type repository* provides a structured storage for type information related to services and interfaces for accessing the services. The web-Pilarcos type repository design was initially born during the evolution of ODP type repository [11] and OMG MOF specifications [12]. Operations are provided for publishing new types, comparing types, and creating relationships between types.

The most essential target concept to be managed in type repository system is *service type*. Service types are abstract descriptions of business service functionality. Services as considered as self describing independent components, as in Service Oriented Computing approach [13]. Service descriptions provide information to ensure technical connectivity, semantic interoperation and behavioural compatibility in possibly heterogeneous environments. At the same time, service descriptions should not expose internal properties of services as this decreases the possibilities of reuse and evolution of services. Implementation specific information, such as binding of a service into specific communication protocol or address, is not covered by service type. Service type is intuitively like an abstract interface or a contract, which an actual service must implement.

Service types are described using a XML-Schema [14, ?] which includes an interface description as embedded abstract WSDL-document, service attributes and an interface protocol. Service interface syntax in web-Pilarcos is described using an abstract WSDL description which does not contain the implementation specific `binding` or `service` -elements (see [15]). Service behaviour is defined as the externally visible message exchange pattern: what kind of messages are received or sent and in what order. Each service supports only one kind of behaviour; different behaviour implies different service type. We refer to the definition of service behaviour as *interface protocol* which is a process-like behavioural description that defines valid behaviour at one endpoint of a bilateral communication. Interface protocols in web-Pilarcos are based on the notion of session types, introduced by Honda et al. [16] and Gay and Hole [17]. For behavioural descriptions we have a simple XML-based process descriptions language, which has WSDL operation calls and labeled branching as communication primitives and session establishment, parallel composition, name hiding and conditional branching as structuring constructs [18]. Semantic interoperability of services is supported by binding ontological concepts to the exchanged documents. XML-based ontology description languages, such as general purpose description languages RDF(S) and OWL [19, 20] or more specialized XML-based ontologies such as RosettaNet, can be used [21].

Service types are published by institutions responsible for a business domain or by enterprises willing to promote use of new kinds of services. Formal standardiza-

tion of a new service type is however not necessary because the applicability and adoptance of the service type is determined by peer acceptance.

The structuring rules of the type system that captures behavioural session types, supports structural matching of syntactic information and supports semantic relations based on description logic [16, 22, ?]. Subtyping-like relationships that support service evolution are also important: solutions for syntactic, behavioural and semantic service typing have been reported [23, 17, ?].

The type discipline in web-Pilarcos platform is strictly managed. Every type definition must be contained by a type repository. Each type name, i.e. URI, must also identify the type repository responsible for managing the corresponding namespace and its type definitions. Without strict management of typing information it would be impossible to ensure that types are unambiguously named, persistently stored, verified to be correct, and relationships between types verified and intact [7]. Type repositories can also be organised into a hierarchy for partitioning of namespaces (compare to DNS [24]).

The *service offer repository* refers to services like UDDI [25] and ODP trading service [26]. The purpose is to locate services that are published using structured metainformation description of the service. In the web-Pilarcos environment, these descriptions are essentially considered as binding offers for the service. In addition, when a new service offer is published, type repository functionality is used to validate the conformance between the offer and the corresponding service type. If the validation is successful, service offer is published into a service offer repository with the claimed service type. The service offer publishing process requires predefined service types.

5 Verification and observation of interoperability

The web-Pilarcos middleware aims for maintaining correct collaborative behaviour in eCommunities, involving several aspects of interoperability requirements. The requirements cover technical, semantic, and pragmatic aspects, i.e., awareness of collaborative behaviour and policies. Traditional verification and static analysis methods are complemented by dynamic observation of behaviour conformance against the contracted BNM and policies.

The research and prototype building in the web-Pilarcos project focuses on interoperability and eCommunity management problems at the business service level, i.e. at the level of eCommunity, its participants, behaviour and lifecycle. As we presume that services are implemented or wrapped using Web Services technology, technical interoperability at the lower protocol levels is well provided by a service oriented technical middleware layer.

Interoperability problems in software systems stem mainly from components' implicit and incorrect assumptions about behaviour of their surrounding environment [27]. Every aspect of service and eCommunity functionality must be made explicit using unambiguous notations. Concepts of compatibility and substitutability are key issues in integration of autonomous services into communities; descriptions of services and communities must therefor be founded on formal basis.

When an eCommunity is established, we ensure *sufficient* conditions for interoperability of services during service discovery and population. If everything goes as planned, the eCommunity will meet its business goal. This however does not mean that during the operation of the eCommunity nothing can go wrong. Participants of a eCommunity may behave incorrectly due to outdated service descriptions

(because of autonomic evolution of services), changed business policies or technical problems. To overcome, or at least identify, interoperability problems during operation of communities we have adopted an approach based on runtime monitoring of eCommunity contracts.

The sufficient conditions for an interoperable eCommunity are fulfilled by three solutions. First, the use of a verified BNM as a basic structuring rule for the eCommunity; the various business process models intertwined into the network model can be verified to be for example deadlock free and complete by traditional protocol verification tools. Second, the use of constraint matching for accepting service offers to fulfill roles in the BNM. And third, the augmentation of the constraint matching process by the interference of further constraints arising from the selected offers for neighbour roles.

Relevant issues in role related constraints cover interface syntax with behaviour descriptions, syntax of documents to be exchanged, semantical aspects of control and information flows, and nonfunctional aspects like trust and business policies that further restrict the behaviour.

To promote evolution of syntactic structures of services, we will adopt principles of by-structure matching instead of by-name matching for service interface comparisons [28]. For syntactic matching we map structures of WSDL and XML-Schema languages to function-, product- and union types and their compositions as described above. Using structural typing constructors for WSDL and XML-Schema definitions we can decide if two WSDL interface descriptions are structurally equal. This interface matching is done using a approach similar that introduced in [29, 22], that is, using bisimulation relation of term automata representing interface structures.

Useful for information exchange are the types of XML-Schema. The web-Pilarcos platform uses built-in types as primitive ground types (see [**?**]) and also maintains complex type definitions and their inter-relationships. Complex type constructs are interpreted as finite commutative product and union types [30] when structural equivalence of documents is validated. Grouping structure `sequence` may also be interpreted as an ordered product type if more discriminating identification of structures is needed.

Service selection and matching based on semantic concepts is not addressed in the present version of the web-Pilarcos platform but it will be implemented in future versions. Matching of semantic concepts shall be implemented using standard theories and tools, such as described in [31]. Meanings for exchanged messages are described with semantic extension attributes of WSDL messages as proposed for example in [32].

Behavioural interoperability is considered in the extent of verifying that service offers and role requirements for service behaviour match, as discussed in Section **??** for accepting service offers to the repository.

We even do not seek to completely prove that a eCommunity behaves correctly, as this would need verification of behaviours between every possible participant in a eCommunity during its establishment process. Even in theory, a complete pre-operational verification of a eCommunity behaviour would be impossible, because of dynamic changes in the system, such as evolving business policies. Instead service types are considered as contracts, and the subtyping of session types as proof of conformance. Inevitable behaviour and policy conflicts are observed and acted on during operational time, by the monitoring system.

Although the monitoring system will be discussed in further detail in an accompanying paper [33], some comments on the issues and effects of detected breaches.

As the monitoring system can be given a fairly free set of rules to monitor passing message traffic, different informational and behavioural aspects are fairly straightforward to monitor. The challenges arise from performance penalties and timeliness.

Some breaches that can be detected by monitoring include a) messages from parties not partners in the eCommunity; b) transactions that are not acceptable in the current state of the eCommunity lifecycle or not fulfilling precedence requirements; c) information contents is not allowed to be exchanged (e.g., private documents, unknown structure); d) expected flow of information is broken; and e) obligatory transactions are not performed.

Each administrative domain can have their own decision method on how critical a breach is considered. The eCommunity contract provides methods for BNMAs to invoke in case of breaches, either for information only, or for the removal of the partner in fault. The eCommunity contract must carry rules for deciding which action to take, and what recovery process to use. In addition, the recovery and sanction processes have to be part of the contract initially.

6 Conclusion

The web-Pilarcos approach supports autonomous services to form federated communities. Federated approach means that there is no overarching shared collaboration model from which the services would be derived. Instead, the services stand on their own and interoperability from collaboration process, semantic and technical view must be maintained explicitly. The web-Pilarcos middleware provides some tools for this, so the service implementors need not to worry about these aspects.

From the BNM, it would be possible to use the popular model driven approach and generate applications conforming to the abstract models. However, the model driven approach is more suitable for implementing integrated solutions. But, those are not resistant for evolution needs: technical computing platforms change; services provided by enterprises change; business policies change; and business processes change. For individual services, the MDA style production process is applicable, as long as the application architecture is appropriate. This further discussed [34, 35].

The federated approach has been criticized for the lack of advise for service elements to be developed. However, making existing business network models globally available and thus exposing repeating patterns of roles - i.e., expected local business processes - gives required guidance. Such publishing has already taken place with RosettaNet etc; our solution is to provide a repository for external process descriptions that can be augmented on demand, and that will provide an element of evolution support. These model definitions can be added to the repositories at will, without interfering already operational communities. Existing models can be frozen so that new communities are not any more formed using them, but are not actually removed automatically. The verification and matching hierarchies within the repositories may depend on them, and of course, operational communities may do references.

An other criticism frequently arising is the performance penalty of the eCommunity interoperability checking. From our earlier prototype on the populator process, we can judge that the cost of the process and its scalability are acceptable [36]. We compared the cost of population process with discovery of all partners, the subsequent negotiation process and the eCommunity termination sequence to a traditional systems where name service is used to discover partners. The additional performance cost was around 10 % (150 milliseconds), dominated by the eCommunity establishment phase (nearly 100 milliseconds). The measurements were done

with a CORBA components based environment with MicoCCM (C++ implementation) and OpenCCM (Java implementation) The significance of this cost depends on the length of the eCommunity life span. Since those measurements we have an improved implementation that is more directed towards interoperability testing needs and has an improved algorithm for propagating additional constraints rising from already selected participants' properties. Further needs are in extending the repositories described in this paper, and in distributing those repositories. Current work focus is on the monitoring system.

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 web-Pilarcos project is a member in national ELO program (E-Business Logistics) [37]. The work much integrates with RM-ODP standards work, and recently has found an interesting context in INTEROP NoE collaboration.

References

1. Afsamanesh, H., Garita, C., Hertzberger, B., Santos Silva, V.: Managment of distributed information in virtual enterprises - the PRODNET approach. In: Proceedings of ICE'97 - International Conference on Concurrent Enterpriseing, Nottingham, UK (1997) http://www.uninova.pt/p̃rodnet.
2. Rabelo, R., Camarinha-Matos, L.M., Vallejos, R.V.: Agent-based brokerage for virtual enterprise creation in the moulds industry. In: E-business and Virtual Enterprises. (2000) http://gsigma-grucon.ufsc.br/massyve.
3. Camarinha-Matos, L.M., Afsarmanesh, H.: Service federation in virtual organisations. In: PROLAMAT'01, Budabest, Hungary (2001)
4. Lazcano, A., Alonso, G., Schuldt, H., Schuler, C.: The wise approach to electronic commerce. International Journal of Computer Systems Science and Engineering (2000)
5. Alonso, G.: Wise: Business to business e-commerce. In: Proceedings of the 9th workshop on research issues on data engineering (RIDE-VE'99), Sydney (1999)
6. Neal, S., Cole, J.B., Linington, P.F., Milosevic, Z., Gibson, S., Kulkarni, S.: Identifying requirements for business contract language: a monitoring perspective. In: EDOC2003. (2003)
7. Kutvonen, L.: Trading services in open distributed environments. PhD thesis, Department of Computer Science, University of Helsinki (1998)
8. ISO/IEC JTC1: Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing. ODP Enterprise Language. (2003) IS13235.
9. Allen, R., Garlan, D.: Formalizing architectural connection. In: Proceedings of the Sixteenth International Conference on Software Engineering, Sorrento, Italy (1994) 71–80
10. Fournet, C., Gonthier, G.: The Join Calculus: A Language for Distributed Mobile Programming. In Barthe, G., Dybjer, P., Pinto, L., Saraiva, J., eds.: Applied Semantics: Advanced Lectures. Volume 2395 of Lecture Notes in Computer Science. Springer Verlag (2002) 268–332
11. ISO/IEC JTC1: (Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing. ODP Type Repository Function) IS14746.
12. Object Management Group: Common Facilities RFP-5: Meta-Object Facility. (1998) OMG TC Document cf/96-05-02.

13. Papazoglou, M.P., Georgakopoulos, D.: Service-oriented computing. Commun. ACM **46** (2003) 24–28
14. W3C: Extensible Markup Language (XML) 1.0. 2 edn. (2000) W3C Recommendation.
15. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web Services Description Language (WSDL) 1.1. W3C. 1.1 edn. (2001)
16. Takeuchi, K., Honda, K., Kubo, M.: An interaction-based language and its typing system. In: 6th European Conference on Parallel Architectures and Languages. (1994) 398–413
17. Gay, S., Hole, M.: Types and subtypes for client-server interactions. Lecture Notes in Computer Science **1576** (1999) 74–90
18. Honda, K., Vasconcelos, V.T., Kubo, M.: Language primitives and type discipline for structured communication-based programming. In: Proceedings of the 7th European Symposium on Programming, Springer-Verlag (1998) 122–138
19. W3C: RDF Vocabulary Description Language 1.0: RDF Schema. (2004) W3C Recommendation 10 February 2004.
20. W3C: OWL Web Ontology Language Guide. (2004) W3C Recommendation 10 February 2004.
21. : Rosettanet implementation framework: Core specification v02.00.00 (2004) http://www.rosettanet.org/.
22. Jha, S., Palsberg, J., Zhao, T.: Efficient Type Matching. Lecture Notes in Computer Science **2303** (2002) 187–206
23. Di Cosmo, R., Pottier, F., Rémy, D.: Subtyping recursive types modulo associative commutative products. Unpublished draft manuscript (2003)
24. Mockapetris, P.: Domain names - concepts and facilities. Technical report, Internet Engineering Task Force (1987) RFC1034.
25. OASIS consortium: UDDI version 3.0 published specification. (2002) http://www.uddi.orgspecification.html.
26. 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.
27. Garlan, D., Allen, R., Ockerbloom, J.: Architectural mismatch or why it's hard to build systems out of existing parts. In: Proceedings of the 17th international conference on Software engineering, ACM Press (1995) 179–185
28. Ruokolainen, T.: Component interoperability. Master's thesis, Department of Computer Science, University of Helsinki (2004) In Finnish.
29. Palsberg, J., Zhao, T.: Efficient and flexible matching of recursive types. Inf. Comput. **171** (2001) 364–387
30. Pierce, B.C.: Types and Programming Languages. The MIT Press, Cambridge, MA (2002)
31. Sriharee, N., Senivongse, T.: Discovering Web Services Using Behavioural Constraints and Ontology. In Stefani, J.B., Demeure, I., Hagimont, D., eds.: 4th Int. Conference on Distributed Applications and Interoperable Systems. Number 2893 in Lecture Notes in Computer Science, IFIP TC6, Springer-Verlab (2003)
32. Peer, J.: Bringing together semantic web and web services. Lecture Notes in Computer Science **2342** (2002) 279–291
33. Kutvonen, L., Metso, J., Ruokolainen, T., Haataja, J.: Collaboration management in dynamic business networks. (2004) Submitted manuscript to INTEROP-ESA.
34. Kutvonen, L.: Relating MDA and inter-enterprise collaboration management. In Akehurst, D., ed.: Second European Workshop on Model Driven Architecture (MDA), University of Kent (2004) 84–88
35. Kutvonen, L.: Challenges for ODP-based infrastructure for managing dynamic B2B networks. In Vallecillo, A., Linington, P., Wood, B., eds.: Workshop on ODP for Enterprise Computing (WODPEC 2004). (2004) 57–64
36. Vähäaho, M., Haataja, J.P., Metso, J., Suoranta, T., Kutvonen, L.: Pilarcos prototype II. Technical report, Department of Computer Science, University of Helsinki (2002) Draft, to be published in 2003.
37. TEKES: ELO program. (2003) http://www.tekes.fi/programs/elo.