Managing Virtual Organizations with Contracts

Janne Metso and Lea Kutvonen

Department of Computer Science
University of Helsinki, Finland
Email: { Janne.Metso | Lea.Kutvonen }@cs.Helsinki.FI

***Abstract*— Electronic business networks are formed by business application services provided by autonomous enterprises. For controlling the business network behaviour and partnerships, electronics contracts are commonly used. This paper focuses on dynamic virtual organizations where participants can join and leave, or be removed, for various reasons, and the organization structure can change for various reasons. A prototype B2B middleware solution is introduced for managing virtual organization contracts that govern aspects ranging from business aspects to technology issues; local middleware services at the enterprise ICT systems are expected to reflect between the contract and the actual runtime system. The contract carries information about the expected participant roles, services, and properties, as well as the expected collaborative business process between participants for monitoring the conformant behaviour at each phase. In addition, the contract covers commitments for recovery from unacceptable situations and access to methods for introducing changes to the organization structure.**

I. Introduction

The present challenge for collaboration across enterprises is to form electronic business networks, or virtual organizations. This is a step forward from tightly-coupled, integrated solutions where inter-enterprise networks are formed case-by-case, and involving significant establishment and maintenance cost for participants.

The current challenge is to form dynamic communities, virtual organizations, where participants can be chosen from fairly open markets: from those making appropriate services available, trustworthy as partners, and interoperable in communication technologies, information semantics, and expected collaborative processes.

When dynamic control for any constellation is needed, the common technique is to introduce a layer of metainformation to model the computing entities and their communication. This model is then made explicitly available during the operation, and operations are provided for making changes to the model.

In the web-Pilarcos middleware architecture, the same principle is used. The virtual organization structure, its partners, and collaborative behaviour (functional and non-functional aspects) between partners is explicitly stored into a distributed contract. The B2B middleware designed and prototyped provides facilities for

- negotiating the contract,
- ensuring static interoperability properties of the suggested virtual organization,
- preparing and establishing necessary control structures for the formed community,
- monitoring the correct behaviour within the community during its lifetime,
- changing members of the community,
- moving the community to new phases of operation, i.e. epochs, where the structure of the organization can change, and
- terminating the virtual organization.

The web-Pilarcos middleware architecture have been described elsewhere [1], but this paper gives a more detailed look at the implementation of the essential operational environment services that make dynamic aspects of the virtual organizations possible. The facilities provided for gathering metainformation and reasoning about the metadata include the following services that can be seen as general infrastructure services provided by trusted third parties, or taken care of more local arrangements:

- business network model repository, where alternative community structures are defined; the network models are expressed in terms of roles, requirements (service, other properties) for role fulfillers, and information exchanges between roles;
- service offer repository, where service providers announce accessibility and properties of their services;
- service type repository, where service type names and associated properties are defined for the use of service trading and service offer matching against the business network models.

Each of these repositories accept metainformation only through a static verification process. Furthermore, the repositories must be distributed/partitioned appropriately to provide a reasonable load balancing.

This paper concentrates on the features of contracts and the B2B middleware that support changes in the virtual organization structure. The contract carries information about the expected participant roles, services, and properties, as well as the expected collaborative business process between participants for monitoring the conformant behaviour at each phase. In addition, the contract covers commitments for recovery from unacceptable situations and access to methods for introducing changes to the organization structure. Section II discusses requirements for contracts, bringing up the needs for business strategies and technical interoperability. Section III outlines the middleware services in the web-Pilarcos prototype. Section IV addresses virtual organization evolution and its management.

Section V gives some contrasts to related work.

II. Contract Requirements

The contract governing a virtual organization has to capture aspects of collaboration at multiple levels [2]. The community and its participants are supported by middleware level services that take the responsibility of being aware of the interoperability for involved business processes, preservation of semantics in exchanges of information, and technical interoperability while communicating.

The contract is a logically central element that can be used to establish and monitor interoperability between independent participants, and that can be considered to be impartial and thus able to protect interests of all participants and the community itself.

The contract must have a degree of enforceability in respect of the participating services. Aspects to be controlled include service behaviour (and application level protocols used), communication channel services (architecture and parameters), and non-functional aspects of the application and communication services, such as QoS.

The service behaviour descriptions for each role within the virtual organization are recorded into the contract, and thus can be used for monitoring whether actual behaviour is conformant. Failures to comply are reported to all participating organizations, and a decision process is initiated to decide what recovery actions need to be taken. The corrective or sanctioning processes can involve more than one partner in the virtual organization, or participants outside of the virtual organization. These processes are an integral part of the business network model; the business processes (seen as application level protocols) can be verified in terms of their recovery potential before associated with breach management processes.

Communication channel descriptions are needed to ensure low level interoperability between service applications. Channel descriptions prevent mismatches that are caused by using different transport layer protocols. These descriptions must be defined for each connection between services and must allow different channel types between different applications. For example communication between application A and B could use web services and communication between application B and C could use CORBA. Part of the communication channel descriptions are transactional and security requirements. These help to provide advanced properties to the communication and require minimal impact on the service application. Secure communication allows virtual organization to conduct business processes with minimal intervention from malicious bystanders. Security measures include authentication of service use and communication encryption. The last important information derived from communication channels are service endpoints for service applications which help to prepare execution platforms and services implementations for virtual organization execution.

The non-functional aspects to be enforced form a set of semantically wide range of elements (from business strategy needs, business values, and trust, to technical aspects such as security and QoS). These aspects can be defined as rules to be monitored during the virtual organization lifetime.

The technical level of interoperability captures the messages used between service applications, and thus, the information encoding and data formats used in passing information between organizations. This is important for syntactical parts of the interoperability. The messagetypes help to enforce the contract as they can be used while monitoring the message exchanges between applications to determine if the messages are syntactically correct or not.

In addition to the metainformation about interoperability levels, the contract provides facilities for metainformation - thus, virtual organization configuration - changes. Changes to the virtual organization structure are made through epochs. Each epoch has its own model and changing the epoch moves the organization to a new model. The result is that epoch change can alter the service that the virtual organization is producing. Organization membership changes can be triggerd in contract breach situations and introducing new organizations to the virtual organization requires renegotiations to the contract.

Because the contract reflects the needs of enterprises doing business through computing systems, the accountability of the virtual organization and the contract itself is important. The contract has to be able to capture restrictions on collaboration, based on laws and regulations. However, our implementation is more focused on technical, semantical, and process-aware interoperability, and thus the additional requirements (rules) to be expressed in the business network models are not further discussed here.

For operational point of view the contract must be identifiable. This requires unique identifier for the contract. If the business process defined in the contract can be executed multiple times the contract needs some sort of session semantics and identifiers. On the operational level the identifiers help to keep up the state of the virtual organization and contract enforcement in the form of service monitoring.

III. The B2B Middleware Services

The web-Pilarcos prototype for the main middleware facilities provides services for managing contracts, inter-enterprise communication and monitoring the agreed behaviour determined in the contract. The components providing these services are NetworkManagementAgent, Contract object, and Monitors. As referred to in the Introduction, there are infrastructure repositories for discovering new partners to virtual organizations and other metainformation. The related service components include Populator (for lookup and interoperability checking of service offers to fulfill the roles in a business network model), TypeRepository, ServiceOfferRepository, and BusinessNetworkModelRepository. The services and their relationships are illustrated in Figure 1. The contract is stored inside the ContractRepository and they communicate with each other through the NetworkManagementAgents. The agent, contract, and monitors are discussed in further detail below.

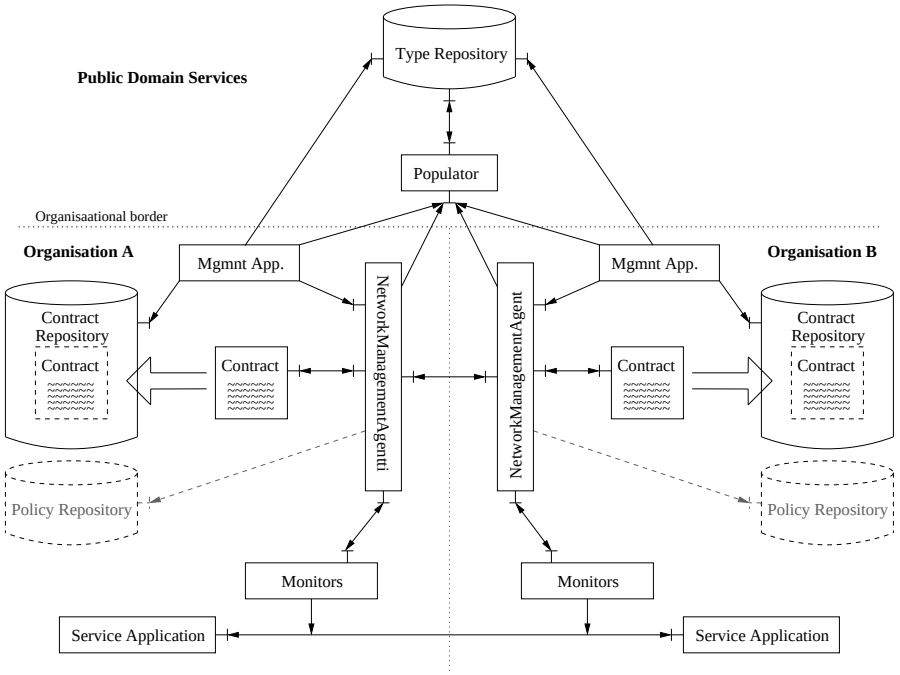

Fig. 1. Web-Pilarcos prototype.

Web services are used as a distribution technology for both inter-organizational and intra-organizational communication. The prototype services described in this paper are all implemented with J2EE [3] technology on top of JBoss [4] platform.

A. *NetworkManagementAgent*

NetworkManagementAgent provides a central interface to an organization. It represents the organization in a eCommunity. Contract objects use agents to communicate with each other during life-cycle transitions and while error recovery. Agents are responsible for configuring the local middleware platforms and services to follow the rules agreed on in the contract. Third role of the agent is to provide a contract life-cycle management interface for internal applications.

The NetworkManagementAgent is implemented as a *SessionBean*. This solution gives good performance and allows multiple threads to execute same code easily. This is important because the agent is used simultaneously by multiple contract objects and multiple partners in different virtual organizations. At the same time local management applications connected to user workstations can be using the life-cycle management operations and planning new virtual organizations.

The protocols for life-cycle management include protocols for all necessary steps. The necessary steps are building, negotiation, execution and management and termination [5]. For each step is a separate protocol. For virtual organization building our prototype uses a BusinessNetworkModel (BMN) based Populator which selects a group of interoperable services to fill the roles of the BNM. The result is returned to the initiator and is only available to the initiator at the first point.

After receiving results from the Populator the initiator chooses the most suitable result for further negotiations. The negotiations are implemented as coordinator driven n-to-n negotiations. The coordinator is initially the initiator but it can be switched to any of the participants during the first round of negotiations or later. During the first round the initiator sends a proposal for the contract to all participants. The proposition is directly based on the Populator response.

After receiving a proposal the participants can either accept the proposal or reject it directly. Another option for a given participant is to make a counter-proposal. If a participant rejects the proposal at any point during the negotiations, a new participant is needed to fill the now vacant role. Finding a new participant requires a new population. The participants can do as many negotiation rounds as are needed for every participant to accept the contract. The negotiations can be abandoned if it seems that there is no result available that is accepted by all participants.

After contract has been agreed on the contract is established. This separate state is added in order to allow for the participants to configure their local platforms and participating services for use in the virtual organization. The coordinator requests the establishment by sending a separate request and when each of the participants have given a response the contract can be executed. Both request for establish and response are simple messages with only contract id as their parameter. After the coordinator has received all responses it broadcasts a message telling that the contract has been established and the virtual organization is usable.

Figure 2 shows the building, negotiating and establishment steps for a virtual organization. First the initiator populates a

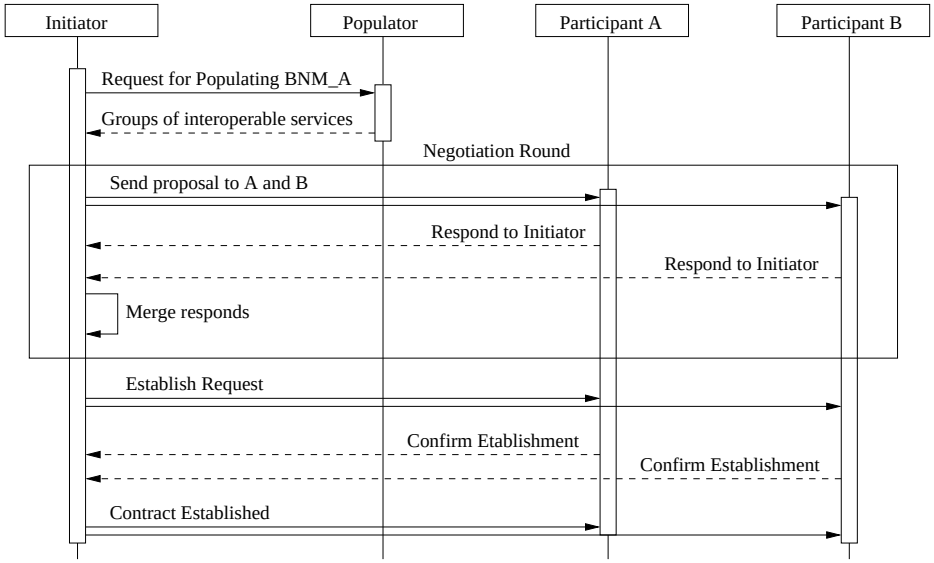

Fig. 2. Building, negotiation, and establishment of the contract.

BNM titled BNM_A using the Populator. After receiving the groups of interoperable services the initiator sends the proposal to organizations A and B. After A and B respond to the proposal the initiator (here selected as the coordinator) merges the results and proceeds to another round. After negotiations have been finished the initiator sends a establishment request to participants and waits for their confirmation. After confirmations have been received the initiator announces that the contract and virtual organization is established and functional.

During the operation of the virtual organization two protocols are used. These are the global state management protocol and breach management protocol. Global state management protocol is used to propagate task level progress of each participating organization and to synchronize and manage epoch changes during the execution. Each organization will report their completed tasks using *UpdateTaskState*-message which identifies the contract, used session, and identifier of the task. Additionally there is information related to identifying the partner such as the signature.

Epochs are changed by sending one message to all other participants when the last of the tasks in the current epoch is finished and the service applications and platforms reconfigure for the new epoch if needed. After sending the message a organization will wait for all other organizations to send the same message. When all of the messages have arrived the virtual organization is ready for the next epoch. In the future when the epoch change will be more complicated (as described in section IV) the protocol needs to be two phased to allow for possible reconfigurations before starting a new epoch such as finding a new participant.

For error resolving there is a protocol to report errors to the coordinator. This protocol is used in situations where the error between two participants is so major that it affects the whole community. Minor errors can be dealt with compensations between the participants. This is only a default protocol and it can be replaced with case-specific protocols during negotiations if desired. The protocol involves a message which is used to report the failure to comply to the agreed behaviour or a failure to produce the required service. The offender can either admit the failure or deny it. To determine if the failure really happened or not a monitoring system (described in subsection III-C) is used. The offender can be removed from the contract if that is voted to be necessary. However if one participant is removed a new participant must be sought by repopulating the BusinessNetworkModel and renegotiating the terms of the contract.

B. Contract Object

The Contract object is a active distributed object in the sense that contract objects communicate with each other to achieve distributed decisions. The object is active because it guards its content and manages its own lifecycle. The contract object is implemented in J2EE as a *EntityBean*. Using EntityBean provides additional robustness by storing the contract related information and the state of the contract object to a database. This means that the prototype is somewhat immune towards server hardware problems.

The contract object encapsulates information described in the section II. For service behaviour descriptions WS-CDL [6] or BPEL [7] can be used. To describe communication channels the prototype uses identifiers which can be used to detect channel types defined in the Type Repository. QoS and other properties appear as name value pairs.

Contact information for the partners is included in the contract. Part of the contact information is organization signatures which can be used to verify that messages indeed are from the organization they are claimed to be sent from. For organization signatures the prototype uses XML Digital

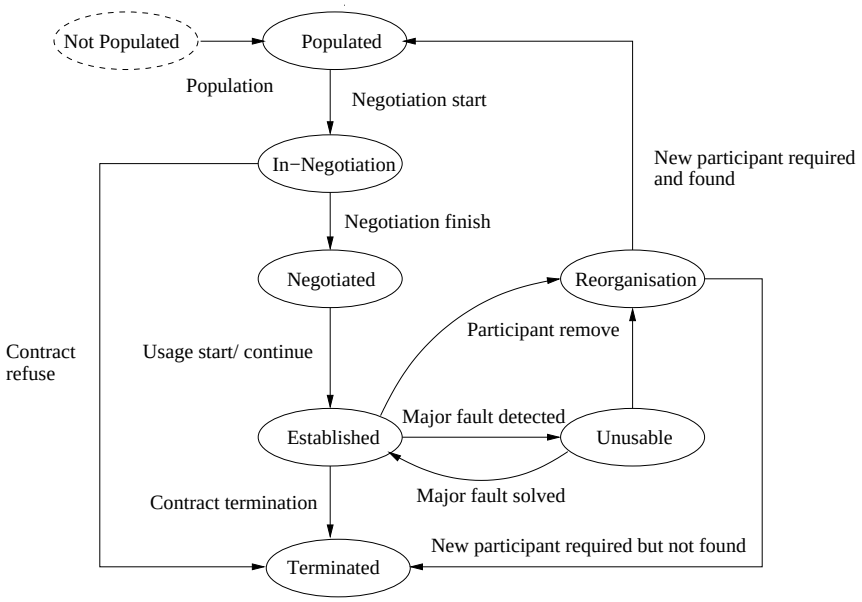

Fig. 3. States and state transitions of Contract.

signatures [8]. Signatures can also be used when encryption keys are negotiated for inter-organizational communication.

Figure 3 describes the contract states and the transitions between states. The states of the contract also determine the lifecycle of the virtual organization. Contract state begins with **populated** and is created from a population response from the Populator by the NetworkManagementAgent. At this point the contract draft is only available to the initiatior (organization who requested the population). The contract enters **In-Negotiation** state when it is sent to all potential participants of the contract. The actual communication between organizations during state transitions are described in subsection III-A. For each negotiation round the participating organization consider the terms of the contract and decide if they are acceptable or not. Any of the participating organizations can terminate the negotiations for their own part at any point it wishes. For organizations who terminated the negotiations the contract will move to **Terminated** state. When the negotiations are finished the contract will globally move to **Negotiated** state. After completing the negotiations there is a short configuration phase for the participating organizations and when all participants are ready the contract will move to **Established** state. During established state the execution of contract is permitted. During contract execution faults in operation can happen and to resolve these situations the contract can move to **Unusable** state. A major fault is defined as a error situation which needs attention from the whole organization such as a failure from one participant to provide required service. When the fault is resolved the contract can return back to established state. If one of the participants leaves during execution or while solving a major fault the contract moves to **Reorganization** state.

In a reorganization case the contract will be repopulated using the original service offers of the remaining participants. This ensures that the new participant will be as compatible as possible to the existing virtual organization. If the repopulation does not return a new participant or it returns the same participant that just left, the contract will move to **Terminated** state and the virtual organization is cancelled. If the repopulation returned a suitable new participant for the virtual organization, the contract will reenter the **populated** state and will be renegotiated using the same algorithm as previously described.

C. Monitoring Service

Monitors provide important support for contract enforcement. Monitors are plugged to part of the communication channel and are used during the contract execution. Monitors are used to follow the service behaviour and its compliance to the behaviour agreed on in the contract.

Monitors are implemented as a mix of *EntityBean*s and *SessionBean*s. The state of the monitored business process is stored as EntityBeans to database and processing of the monitored business process and observed messages are implemented as SessionBeans. Monitors connect directly to NetworkManagementAgents and monitoring metadata is configured to the monitors by the NetworkManagementAgent.

The monitoring algorithm is based on a matrix representation of the business process state machine where the business process states and transitions between the states are coded into a matrix [9]. The matrix is then used to verify is a message allowed to be exchanged between the applications during a certain time. Into this matrix can be coded both the different execution paths and the required message types in these paths. When a message is detected the monitors traverse the matrix to find the current state of the business process and determines which messages can be used to move forward in the process. If the message is not any of the required types, sending the message is not conformant with the service behaviour.

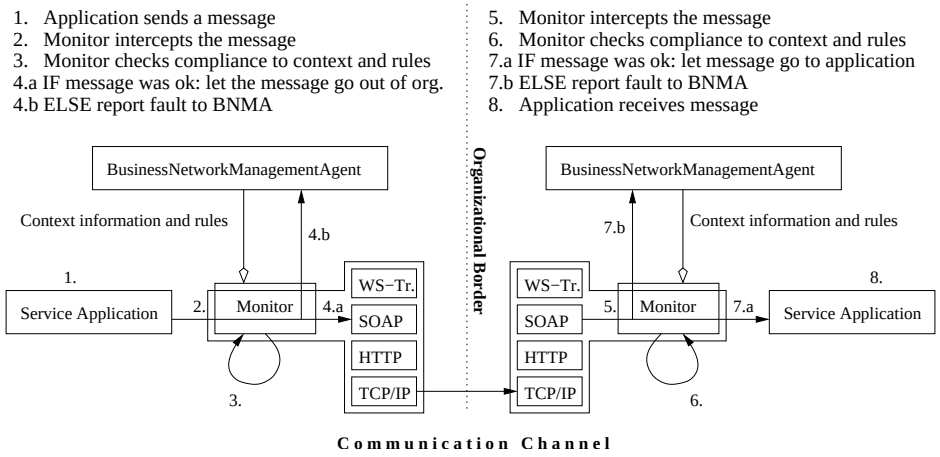

Fig. 4. Monitor as a part of communication channel.

The monitor can be configured to be be either proactive, active, or passive. When it is proactive the monitor will actively stop all communication that is not conformant to the service behaviour and report them to the NetworkManagementAgent. With active and passive monitoring the monitor will not intervene as strongly to the application behaviour as with proactive monitoring. The messages are not blocked but with active monitoring the monitors will report non compliant service behaviour to the NetworkManagementAgents. If the monitor is configured for passive monitoring, the monitor will only log service behaviour and conformance to the agreed behaviour can be verified at a later time.

Figure 4 positions the monitor as a part of the communication channel. The context information and monitoring rules for a service application are configure to the monitor by NetworkManagementAgent in both participating organizations. When the application send a message using the communication channel, the monitor intercepts the message and determines if sending the message conforms to the service behaviour description. In Figure 4 the monitor is proactive and prevents the message from going further if the message was not conformable. Monitors in the prototype are placed symmetrically to both ends of the communication channel. The communication channel can consist of further properties and characteristics such as transactional properties as demonstrated by the **WS-Tr.** on top of the **SOAP** layer [10].

When the monitor is configure to be proactive both incoming and out going messages will be stopped. This represents the highest level of contract enforcement in our prototype. Even though proactive monitoring is not extensively used in industry, there is demand for it in rapidly evolving virtual organizations. Further proactive monitoring is very useful when combined with reputation management systems. By using proactive monitoring an organization can prevent negative impact on reputation from badly behaving service applications. However monitoring can not help in situations when the application malfunctions producing unexpected results or fails completely.

Reports of the monitor depend on the configuration as stated above. In addition to fault reports the monitor reports to NetworkManagementAgents based on the progress of the external behaviour. The behaviour is divided into tasks and monitor reports to the agent when a task is started and finished. When the monitor detects a task start or task end it calls `updateEpochState` operation from the agent interface. This operation determines the session in which the message that caused the state change was executed, role of the application, the epoch in which the update was executed and the new state of the task. The states of the task are **not started**, **started**, **finished**, and **failed**. The task based grouping of business process activities is similar to WS-CDL [6].

The reports allow the agent and contract object to be aware of the progress of the whole business process of the virtual organization. This is important because the epoch transitions are synchronized through out the virtual organization. The progress of each participating organizations is shared between all the other participating organizations. The progress is shared on task basis. Only the task end reports are propagated to all other participants.

IV. Managing Changes in Virtual Organizations

Changes in virtual organization structures can occur for two reasons. First, a participant can behave incorrectly and thus a separate breach management process is entered. Second, the virtual organization can transit to use a new business network model. An epoch change is used to synchronize the virtual organization so that all participants do the model change.

In both cases the necessary technical steps to support the changes are fairly similar. The virtual organization needs to be synchronized to determine the effects of, for example, changing a participant and to minimize them. The virtual organization needs to be repopulated, and interoperability checked. After introducing the new participant the contract must be renegotiated in order to reach acceptable terms for all participants.

An epoch change is a well defined point in collaboration where changing of the underlying model is permitted. In contrast, single misbehaving participants can be removed from the virtual organization at any point. A contract can cover a number of epochs depending how many modifications to the virtual organization structure are needed. Epochs are defined in the BusinessNetworkModel during its design.

Changing participants during the virtual organization lifetime requires major support from the B2B middleware. First of all, it is necessary to support repopulation of the contract to find a new partner, renegotiation of the contract with the new partner, and capabilities to adapt to the resulting changes in service endpoints and policies. More difficult are questions on restart or transfer of ongoing business transactions to new, replacing partners. This is particularly challenging because of heterogeneous implementation platforms the state of services is not easily transferable to another organization (if at all). Therefore, the middleware must be able to gracefully terminate affected business transactions and restart them with as little interference to other transactions as possible.

In practice, most of the business transactions between multiple partners are related to or nested with each other. The worst case scenario is that all ongoing transactions need to be rolled back in every participants' IT infrastructure. This poses potentially a considerable cost to changing participant. The issue is highlighted when there are multiple concurrent executions of the contract. These problems can be minimized by designing such BNMs that the business transactions between two participants are as independent as possible from other transactions.

The epoch changes can also serve as a point where the business processes themselves are redefined. The refining can be done in order to improve efficiency or to fix problems in the BNM. Epoch change is also a suitable point to introduce more partners in to the virtual organization. For example in a situation that not all roles was populated during the initial population a missing partner can be introduced. The reason that a partner is missing can be that there was not suitable service provider available or the service is not needed until at a later phase in the contract execution.

Introducing new partners during the execution of the contract business process requires a new round of negotiations to the contract in order to be acceptable for all partners. The BNM must be designed so that the introduction of a new partner to the virtual organization does not interfere with already accepted policies or the impact is minimal. If the nature of the BNM is such that the impact can not be avoided it must be clearly stated in the description so that potential users of the BMN can take it into account.

Concurrent executions of contract business process make the situation somewhat difficult. There is two possible solutions and the first is to introduce a new partner only once when the first execution reaches the point of requiring a new partner. The second choice is to introduce a new partner each time a execution reaches the point of needing a new partner. The second option will lead to increasing complexity and need for negotiations each time a new participant is introduced. The introduced participant can be the same each time but that can not be relied upon. For these reasons the second option should be avoided in favor of the first option.

Releasing partners from the organization can be done during epoch changes in similar manner than introducing new partners. It has the same set of requirements for the BNM and similar concerns for concurrent executions. Releasing a partner before it is certain that a partner is no longer needed causes the remaining portion of the contract business process execution to lack one partner and the virtual organization must be repopulated. This means another round of negotiations to incorporate the new partner to the virtual organization.

V. Related Work

In respect to the contractual concepts, the web-Pilarcos project relates for example to BCA [11]. The BCA contracts have legal and business level focus, while the web-Pilarcos approach ties together ICT related viewpoints of ODP (Open Distributed Processing reference model [12]), also ranging to some features of business aspects. The ODP-RM introduces information, computational, engineering and technical viewpoints. Each of these present interrelated but somewhat independent aspects of the collaboration features and its composition using more basic computing services. The web-Pilarcos contract structure captures these aspects in its BNMs, binding requirements, and behavioural and non-functional monitoring rules [13]. Furthermore, BCA is closely related to an approach [14] with a centralized notary to detect contract breaches post-operatively. The web-Pilarcos approach aims for more real-time intervention.

The main difference between our approach and other agent based approaches such as MASSYVE [15] is the fact that our agents do not provide other services than network management services themselves. The services are provided by separate components. The second difference is that our agents do not manage workflows. The workflow is managed by separate engines or by applications themselves. In contrast to [16] monitors are used to follow the service behaviour instead of agents and monitors report to agents. In our approach the agents have the task of semantic verification and error resolving during the contract execution.

Most virtual enterprise support environments trust on models for distributed business process enactment. However the web-Pilarcos approach leaves enactment as a local business processing task, concentrating on interoperability monitoring. CrossFlow [17] uses contracts as a basis for cooperation management. The key element in the architecture is a matchmaking engine (trader) that matches contract suggestions and requests from potential partners. Based on the specifications in the contract, a dynamic contract and service enactment infrastructure is set up. The architecture involves matchmaking and outsourcing. WISE (workflow-based internet services) [18], [19] addresses process definition, enactment, monitoring and coordination in virtual enterprises. The process definition component allows composition of virtual business processes

from building blocks published by partners. The process model is then compiled for enactment. The process monitor provides information for load balancing, routing, QoS and analysis purposes. Contract-based coordination is also presented by [20], [21].

VI. Conclusion

The web-Pilarcos architecture provides a B2B middleware layer that supports management of virtual organizations. The management facilities are based on shared vision of metainformation captured into a eContract. Changes in the contract are locally reflected to the enterprise computing system; and correspondingly, relevant progress and breach reports are delivered to partners through the eContract.

The architecture follows a federated approach: participating services are independent and pre-existing, and the collaborative behaviour model is used only for watching conformance. Enforcement of the contract is reached through the independent monitoring facilities at each participant. Those monitors basically react to events that should not take place at that service or resource interface. Those self-protective reactions are then used as triggers for corrective actions for the benefit of the whole virtual organization.

The evolution support suggested by the web-Pilarcos project is challenging and has not yet been studied much elsewhere. However, the facilities required for evolution steps are mostly made available by the original breeding environment services.

The major drawback in the architecture is in the lack of trust management: business can only be run based on sufficient trust between organizations getting involved to the joint business network. Therefore, we are in process of intertwining a trust-based interception system designed in the TuBE [22] project with the monitoring system. Trust and reputation based decisions are relevant at virtual organization population phase, and again, at each business process.

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.-P. Haataja, "Interoperability middleware for federated enterprise applications in web-Pilarcos," in *INTEROP-ESA'05*, 2005.

[2] L. Kutvonen, "Automated management of inter-organisational applications." in *Eight IEEE International Enterprise Distributed Object Computing*. IEEE Computer Society, 2002, pp. 27–38, http://www.cs.helsinki.fi/group/pilarcos/deliverables/kutvonen_management_edoc_2002.pdf.

[3] "Java2 Enterprise Edition," Sun Microsystems Inc., May 2004, http://java.sun.com/j2ee/.

[4] "JBoss, J2EE Application Server," May 2004, http://www.jboss.org.

[5] S. Neal, J. Cole, P. Linington, Z. Milosevic, S. Gibson, and S. Kulkarni, "Identifying requirements for business contract language: a monitoring perspective," in *Proceedings of the seventh International Enterprise Distributed Object Computing Conference*. IEEE Communications, 2003, pp. 50–61, http://www.cs.kent.ac.uk/pubs/2003/1807.

[6] N. Kavantzas, D. Burdett, and G. R. et al., *Web Services Choreography Description language*, W3C, Oct. 2004, http://www.w3.org/TR/2004/WD-ws-cdl-10-20041012/, Working draft.

[7] S. Thatte, T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, , I. Trickovic, and S. Weerawarana, *Business Process Execution Language for Web Services*, BEA Systems, IBM, Microsoft, SAP AG, and Siebel Systems, ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf.

[8] M. Bartel, J. Boyer, B. Fox, B. LaMacchia, and E. Simon, *XML-Signature Syntax and Processing*, W3C, Feb. 2002, http://www.w3.org/TR/xmldsig-core/.

[9] J.-P. Haataja, "Automatized monitoring of inter-enterprise interoperability," C-2005-17, Department of Computer Science, University of Helsinki, 2005, in Finnish.

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

[11] Z. Milosevic, P. F. Linington, S.Gibson, S. Kulkarni, and J.Cole, "Inter-organisational collaborations supported by e-contracts," in *The fourth IFIP conference on E-commerce, E-Business, E-Government*, Toulouse, France, Aug. 2004.

[12] *Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing. Reference Model of Open Distributed Processing.*, ISO/IEC JTC1, 1996, iS10746.

[13] 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

[14] G. Quirchmayr, Z. Milosevic, R. Tagg, J. Cole, and S. Kulkarni, "Establishment of virtual enterprise contracts," in *Database and Expert Systems Applications : 13th International Conference*, vol. LNCS 2453. Springer-Verlag, 2002, pp. 236–.

[15] R. Rabelo, L. M. Camarinha-Matos, and R. V. Vallejos, "Agent-based brokerage for virtual enterprise creation in the moulds industry," in *E-business and Virtual Enterprises*, 2000, http://gsigma-grucon.ufsc.br/massyve.

[16] A. Daskalopulu1, T. Dimitrakos, and T. Maibaum, "Evidence-based electronic contract performance monitoring," *Group Decision and Negotiation*, vol. 11, no. 6, pp. 469 – 485, Nov. 2002.

[17] P. Grefen, K. Aberer, Y. Hoffner, and H. Ludwig, "CrossFlow: Cross-Organizational Workfbw Management in Dynamic Virtual Enterprises," *International Journal of Computer Systmes Sciences and Engineering*, vol. 15, no. 5, pp. 277–290, 2000.

[18] C. Schuler, H. Schuldt, G. Alonso, and H. H. Schek, "Workfbws over workfbws: Practical experiences with the integration of sap r/3 busienss workfbws in wise," in *Enterprise-wide and Cross-enterprise Workflow Managment: Concepts, Systems, Applications*, Paderborn, Germany, 1999.

[19] A. Lazcano, G. Alonso, H. Schuldt, and C. Schuler, "The wise approach to electronic commerce," *International Journal of Computer Systems Science and Engineering*, 2000.

[20] H. Weigand and W.-J. van den Heuvel, "Meta-patterns for electronic commerce transactions based of fbc," 1999.

[21] W.-J. van den Heuvel and H. Weigand, "Coordinating web-service enabled business transactions with contracts," in *Proceedings of the 15th Conference on Advanced Information Systems Engineering (CAiSE 2003)*, vol. LNCS, 2681. Springer Verlag, 2003, pp. 568–583.

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