A Concept for Supporting the Formation of Virtual Corporations through Negotiation

Boris Kötting
University of Kaiserslautern
P.O. Box 3049
67653 Kaiserslautern, Germany
koetting@informatik.uni-kl.de

Frank Maurer
University of Calgary
2500 University Dr NW
Calgary, Alberta, T2N 1N4 Canada
maurer@cpsc.ucalgary.ca

Abstract

This paper describes a system that supports software development processes in virtual software corporations. A virtual software corporation consists of a set of enterprises that cooperate in projects to fulfill customer needs. Contracts are negotiated in the whole lifecycle of a software development project. The negotiations really influence the performance of a company. Therefore, it is useful to support negotiations and planning decisions with software agents. Our approach integrates software agent approaches for negotiation support with flexible multi-server workflow engines.

1. Introduction

The term *virtual corporation* is not new and was first discussed intensively by [6] where they describe it as form to organize enterprises. Since then the numbers of publications concerning virtuality in connection with companies was growing extensively and led to some different definitions of the term depending on different views. A good description comes from [4]:

"A virtual corporation is a temporary network of independent companies – suppliers, customers, even erstwhile rivals – linked by information technology to share skills, cost and access to one others markets. It will have neither central office nor organizational chart."

The most important properties of such temporary networks are

- each company brings in its core competence
- relationships are not permanent and less formalized
- mutual dependencies between participants
- heavy use of information technology

All authors agree on the aim of virtual companies: to react faster on changes in the environment of the company.

Confidence and understanding between the corporation partners as well as intensive use of information technologies were mentioned as success factors [6, 7].

A virtual software development corporation is one example for a virtual enterprise. In this paper, we propose an approach that supports software development processes in virtual software corporations. We believe that a virtual software development corporation will have to fulfill the following roles:

- representative

The representative is an entity that acquires customer contracts and is the addressee for the customer for the whole project. From customer perspective, the representative *is* the virtual company: it is the point of contact, signs the development contract, and is responsible for fulfilling it.

- coordinator

Coordination is necessary for software development projects. Coordinator entities are responsible for the well-organized process execution and for enabling collaboration between the participating companies. Coordinators communicate with both, the development companies and the representative.

- software development company

Companies (or parts of companies) collaborate in the actual software development in the project. For a new project, the set of companies that participate in the virtual software corporation has to be determined. This set may change at any time in the course of the project because other competence is needed that can only be provided by different companies. These changes in the organizational structure and the interactions between companies and customer are the reason for describing a virtual corporation as fluid [4].

It should also be mentioned that virtual software development corporations already exist and are successful [15].

Different implementations of this organization model are conceivable.

The most often case is that one company takes the role of the representative, the only coordinator and one

software development company. This company undoubtedly leads the software development process and sources out some parts of the development where it has little competence. This is a rather centralized approach for a virtual enterprise that – according to theory – should be formed by a set of peer companies.

An alternative would be splitting the roles of the coordinator and the representative between two companies. The company with the strongest core competence required for the project would act as the coordinator and lead the development process whereas another company would represent the virtual enterprise against the customer. The representative needs to know the core competencies of the companies participating on the project and then select one company as the coordinator. Then the coordinator and the representative together have to select the companies that should participate in the project.

The best case - according to economic theory - is that a different and independent company performs every role and that there is no leading company in the network. So, one company can act as representative and is responsible for public relation, promotion and recruiting of customers. Other companies coordinate the development process. The software companies participating on the project can concentrate on software development, their core competence.

Existing workflow management systems (WFMS) are typically focused on process support for a single company. They are rigidly relying on the stability of the company structure - which is not the case for virtual software development corporation (VSDC) - and require much work if the structure changes completely. Hence they support the distributed *execution* of a workflow and shared access to data but they do not support virtual corporations at all.

Most of the required planning, coordination, and change impact management functionality required are not fulfilled by traditional WFMS. Approaches like [16] can support cross-organization projects but omit coordinator functionality.

To build a virtual software corporation it is necessary to split up the tasks to be performed between the companies involved. If a task is not executed by the coordinator itself negotiations with other companies need to be carried out. Successful negotiations will lead to a contract. The parties involved in the set of contracts made for the project form the VSDC. A process support system for a VSDC must therefore provide the coordinator with data to help her making decisions for or against possibly participating companies. We want to assist development processes in virtual software companies that follow any of the structures discussed above. Our approach supports

- modeling & planning of software development processes,
- negotiation processes distributing the enactment of task sets to different companies, and
- the execution of tasks using communicating workflow engines

In Section 2, we introduce our current approach and the extensions needed to support virtual software corporations. Section 3 deals with our concept for supporting negotiations using software agents. The state of the implementation is described in Section 4. A discussion of the approach and related work follows in Section 5.

2. Process support for VSDC

In the following, we first discuss our MILOS system. Then we explain extensions that are steps towards a process support environment for virtual software enterprises.

2.1. The existing system

Over the last couple of years, we developed the software process support system MILOS. Its focus was on providing flexibility by interleaving project planning with project enactment and on supporting changes by automatic change notifications.

The major components of the MILOS are the resource pool, the process model, the project plan and the workflow engine (Fig.1). Components are linked by an event propagation mechanism that sends notifications about changes to all observers (other components and users). For a more detailed description see [2,11].

The *resource pool* component manages roles, agents and agent properties. It allows representing the organizational structure of a company as well as hierarchical skill sets. An agent can have a set of skills. The project planner can query the resource pool for all agents with specific skills. The query would result in best matches based on the hierarchical structure of the skill sets.

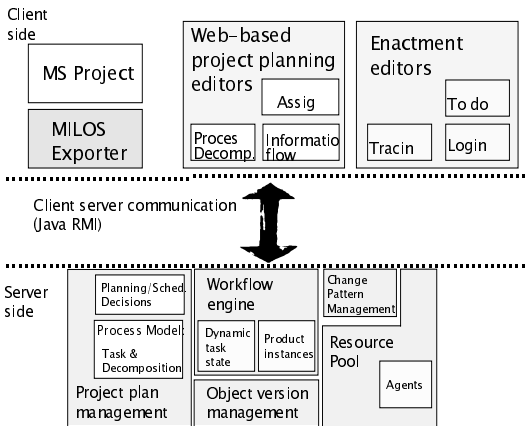

Fig. 1: The MILOS-Architecture

The *process model* component handles process definitions including control flow (pre- and postconditions), information flow, and process decompositions. For every process, we can describe required skills.The *project plan management* component customizing a process model resulting in a project plan. Beside adding/removing tasks, customizing includes scheduling planned start and end times of processes and assigning agents to processes.

The *workflow management* component is responsible for enacting the project plan and managing products. It generates to-do lists for agents and maintains the current state of the project. The workflow engine is able to react dynamically to project plan changes during execution, without interrupting the execution.

Process modeling, planning and execution can be distributed to different companies and areas by running the clients on an arbitrary machine connected to the Internet. Nevertheless, the disadvantage is the centralized structure of the system. For every project, there is exactly one process model, project plan and workflow engine on the main server. In this approach, everybody is working on one model and the possibility for conflicts is very low.[1] Furthermore, security and access to data is handled by a central authority (who runs the server) and relies on a trust relationship between this company and all companies that store data on the server.

A single server may also become a bottleneck and a single point of failure. Latency and bandwidth between a client and the server may not be acceptable resulting in delays in accessing the server.

It is also debatable if companies would join a project executed in such an environment because they do not have control over their data. The server - and therefore their data stored on it - is outside their control. These arguments are psychological but nevertheless relevant. In fast-changing virtual enterprises, trust between participating companies may not be there (at least at the beginning of the cooperation) and companies tend to (or need to) protect their intellectual property.

2.2. A first extension of the MILOS-System

Enactment in distributed network areas leads us to the first step of distributing the enactment component of the MILOS system: the replication of the workflow engine[2]. The main reasons for this replication is to overcome bandwidth problems (at least from the user's perspective) and to increase the reliability of the system.

Our system will create replicates of the workflow engine on several servers distributed all over the world. All engines have the same rights and the same data, although the project starts with one initial workflow engine. To synchronize the state of the project enactment, every engine propagates changes to all other. The replication is transparent to the user. S/he is not aware of the synchronization process running in the background but gets notifications caused by changes in other workflow engines that influence her work.

Every client is connected to his "local" engine – an engine to which a fast Internet/Intranet connection exists whereas there may be only slow connections between engines running on remote servers. From the user's point of view, this results in fast answers by his server.

Another benefit from the replication of workflow engines is the possibility for a company to go off-line. For the time it is off-line, it gets no change notifications but all changes are buffered by the other WFEs. When the workflow engines are reconnected, sending all buffered messages synchronizes them. The users have to resolve any existing conflicts manually after receiving a notification messages about the conflict.

The replication approach as described above has some disadvantages.

First, complete replicates are created requiring that all data be transferred to every environment. This leads to large redundancy. Companies store data about activities and products that they will never use. Moreover, the data transferred could be company-sensitive and so the transfer should be forbidden.

Second, the fact that every workflow engine has equal rights allows everybody to change the model, plan and workflow descriptions. An approach for realizing permissions is necessary for proper development process support and also a role for setting and managing these permissions. Compared with the structure of a virtual software development company described above, this is a task that should be performed by a coordinator.

2.3. A second extension

A first improvement would be an approach where one workflow engine (WFE) is the center and its data is replicated on demand to others. Companies only fetch data they really need. This breaks down the network traffic and the data stored locally but it leaves other problems unsolved.

The workflow engines do not work on *complete* replicates which increases the risk that a user experiences delays because data is fetch on-demand via a slow connection.

This approach offers the possibility for granting access rights for specific parts of the project. The central WFE fulfills the role of a coordinator and leads the execution of the project.

To improve the multi-server, full-replication model, a communicator object is inserted into the WFE design [17]. This object takes advantage of the workflow

---

[1] We use an OODBMS to handle conflicts resulting from concurrent access to the data.

[2] We focus here on the replication of the workflow engine because it is the central component for process support. Clearly, to replicate the WFE, we also have to replicate its underlying data structures (project plan and process model).

replication mechanism described above and acts as a filter for messages to be send to other companies[1]. Additionally, it avoids broadcasting of changes by sending messages only to workflow engines that are affected by a change. The leading workflow engine reflects the coordinator role of a virtual software development company. The communicator object talks to the local workflow engine as well as to communicator objects of all other WFEs. This assumes that it knows all other companies involved in the virtual enterprise. The approach results in a star-like communication structure.

Besides forwarding and filtering messages the communicator object needs to support the initialization of a partial project plan in a different company. It has to contact the communicator object in that company and transport a partial plan to the other location.

Planning support is improved by expanding the project plan component. Imagine a process model that was tailored by adding planning decisions and scheduling. The project plan forms a decomposition tree reflecting process refinement from the root node to leaves. For every node in the tree the responsible planner can transfer the execution of a process to a different company by marking it as external and assigning it to another company. Automatically, all subprocesses of the external process are also marked as external. Furthermore s/he can set the right for changing project plan and process model. That leads to several possible relationships between the two companies. If the right to plan the project stays with the coordinator, it is an outsourcing relationship. The company acting as contractor can only state their change suggestions to the coordinating company that makes the decision of accepting or refusing them. If the right to change the plan is given away the contractor can be make changes. He only needs to make sure that the correct results are produced.

Sometimes the aim of the coordinator is to completely guide another company but in other cases the results of a process might be the only point of interest. If the later is the goal, it is useful to omit the planning at all and to mark a leave node of the project plan as external. For the coordinating workflow engine this planned process is like a stub. The workflow engine in the other domain needs the change right to refine the node and create an own (sub-) project plan.

This approach is still restrictive. Communication is routed via the central coordinator. The economic reality shows that the coordinator normally would also be a software developing company itself. So, this approach is most suited for an outsourcing structure.

One major aim of lean management, which is one of the reasons for creating a virtual enterprise, is to flatten communication structures. Concentrating the control flow and data flow on one company generates an artificial bottleneck. In many cases it would be best if two companies would communicate directly to solve a problem without involving the leading company. Probably, the leading company has no interest in controlling every little step of the development process.

[1] Specifically: To the workflow engines in other companies.

2.4. A third extension

Ideally, the communication links between the workflow engines of the companies involved need to be peer-to-peer (Fig. 2). Communicator objects send messages to the local workflow engine as well as to communicator objects of all other companies. This results in a network of communication links.

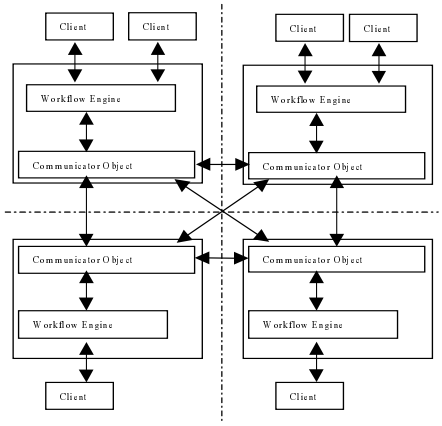

Fig. 2: The third extension of the MILOS-System

In extension 2 only the leading workflow engine has knowledge about all companies and knows when to inform which communicator object about changes. All other engines only address the central communicator.

3. Introducing negotiation agents

Consider the following scenario. A customer announcement regarding a software product leads over negotiations to a software development contract with the representative. This contract will be forwarded to the coordinator who plans the project. There are different ways to determine the tasks that need to be done [10]. We focus on architectural considerations (which are probably the most often used criteria). After the system architecture parts are defined, the virtual corporation needs to be formed by starting negotiation to sign contracts that guarantee a successful enactment of the software development process.

For supporting negotiation, we provide the concepts of software agents. The agents have no common goal but each agent pursues his own agenda and interests, either competing or collaborating with other agents. In a first step, the agents are semi-auto mated by contacting a human agent when a negotiation is offered. There are mainly two points we want to look at in this section:

- how can agents interact with each other

- what is the format for the bidding process to get a contract

3.1. The contract net protocol

The Contract-net is a negotiation protocol proposed by [14]. It provides a model how agents can interact in negotiations.

An agent with a task to offer broadcasts a call for bids and waits for replies for some time. After this time elapsed, it awards a contract to the best bid (according to its selection criteria).

This protocol has been widely used and there are some expansions of that protocol like [13].

3.2. Negotiation requirements

In the easiest scenario of negotiations, a contractor offers an *announcement* to another agent and waits for a positive or negative reaction. If the contractee accepts the offer, the contractor needs to commit the contract to make it valid (Fig.3).

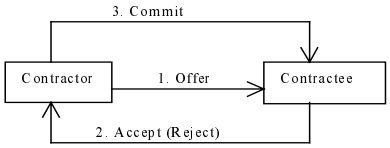

Fig. 3: Basic Commitment

But negotiations aren't that easy today. Usually the announcement will be send to a set of agents. There are many ways for determining potential agents for the broadcast. In our first realization, every agent is able to announce himself (respective its core competence and, optionally, available resources) to other agents.

An agent accepting an offer has no guarantee that he will get the contract (Fig.4). A contractor can reject the offer or may not even react at all. The same holds for the contractee: he may not even respond to the offer. For this reason, broadcast offers have a timeframe in which a reaction is excepted. If this time has passed, the contractor decides which contractee will get the contract[1].

Often the payment for a contract is open and can be negotiated, too. Contractors now have to *bid* to get the contract. This is a more complex situation for the agent because it needs to calculate the costs (which could be very difficult, especially without existing data). This calculation can also include outsourcing of parts of that contract. So, it can directly put the contractee into the role of a contractor.

Awarding the contract is a complex function and will not be automated in our system. The decision for a contractee not only depends on the monetary best bid but also on past experiences with the contractee and other contract conditions. If this information is available, the computer agent will provide it to the human agent.

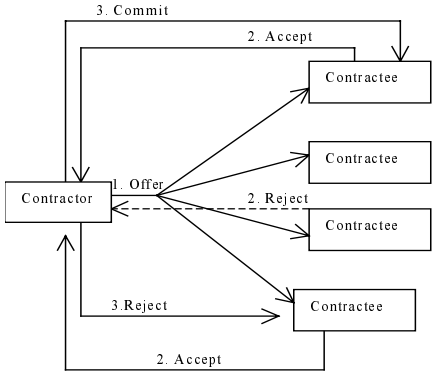

Fig. 4: Broadcast Commitment

Another enhancement of the negotiation is the possibility for the contractee to make a counterproposal. This is for example the case if

- the contractee cannot perform the whole contract but parts of it (he has not the needed resources to perform the whole contract)
- the contractee does not want to perform parts of the contract (because there are parts that are beyond its core competencies)
- the contractee wants a different amount of money or a different schedule

The contractee can offer a modified contract to the contractor, waiting for its reaction. The contractor can also make counterproposal to this contract and so on. It has to be mentioned that any company involved in this process cannot retreat from an offer it has made in a counterproposal.

This enhancement clearly complicates the process of awarding the contract to a contractee because the responsible human agent has a lot more options. Especially if his offers gets no direct accept, he needs to consider the different offers carefully.

3.3. An negotiation support approach

From the requirements above, we derive the following structures for our software agent support for negotiation.

a) The announcement structure

The negotiation about a set of possible tasks can consist of some iteration. The terms of the contract may change but are still based on the first offer. Version management is necessary to clearly identify what agent makes which bid or counterproposal. Therefore, the structure we propose has a *negotiation part* and a *contract part* for the contents.

[1] We do not realize the contract net concept of penalties because it seems not be realistic in our application setting.

For negotiation purposes, we decided to include following information:

- The identifier for the sender of the message
- The announcement identifier
- The version number of the announcement

With this information, a unique identification of an announcement is possible.

The structure of a contract is based on [10]. We expanded it to fulfill the demands of our model. A contract describes:

- Tasks or activities to be performed
- Deliverables (code, documentation, tools to be used)
- Standard or methods to be followed
- Schedules for tasks and deliverables
- Type of equipment to be used (development platforms, target platforms, test tools)
- Miscellaneous items (special interfaces, performance requirements)
- Partial project plan that need to be performed (optional)
- Cost structures (There exists several static and dynamic payment models)
- Capital equipment
  Describes if the subcontractor will have access to capital equipment (i.e. hard and software) and in what size this access will be
- Warranty
- Conditions for payment (For example, payment can be linked to milestones or paid after complete delivery).

a) Software agent structure and activities

Every agent has access to the set of tasks that need to be performed and access to the set of resources available to handle tasks (including financial resources). These sets vary during project enactment because resources are needed for performing tasks and new tasks come from other agents. Also tasks can be outsourced. Furthermore, the agent manages a list of announcements that include the history of its bidding (proposal) process.

As a first approach, agents can offer to subcontract tasks to other agents by paying money, the amount depends on the contract terms. Agents offer tasks for reasons of profitability: subcontracting of a task is profitable, when the subcontractor can handle the task cheaper as the contractor or if the contractor cannot perform the task at all (e.g. because of missing resources).

Every software negotiation agent can act as a contractor and as a contractee. Hence he needs to provide the following functionality:

- methods to create and edit a contract
- methods to administer existing and incoming contracts
- methods to access resources and tasks
- methods to manage information about other negotiation agents
- methods to provide support to the connected human agent that finally makes the decision (i.e. provide negotiation history)
- announce capabilities or introduce itself to a set of agents
- announce contracts to a set of agents
- send a counterproposal to a contractor
- send a counterproposal to a contractee
- make a bid to the contractor
- send accept/reject message to a contractor
- commit a contract to contractee

To commit/reject/accept a contract, the agent only needs to transfer the announcement-id and the version-id introduced above. For a counterproposal or a bid the agent needs to transfer the new announcement object to the receiver.

4. State of implementation

A single-server version of the MILOS system will be demonstrated as part of the formal research demo track on ICSE 99 in LA in May 1999. The implementation of extension 1 (complete replication of WFE) and extension 2 (lead WFE plus independent client WFEs) is under way and is planned to be finished early summer 1999.

MILOS is implemented in Java. We are using the OODB GemStone/J 2.0 as an Enterprise Java Bean (EJB) server that provides transaction management and persistency services. EJB is a portable, highly scalable, multi-platform component architecture that dramatically simplifies the development of thin-client, multi-tier applications. The clients of our system are Web-based applets using Java Remote Method Invocation (RMI) to communicate with the EJB server.

5. Discussion and Related Work

Several areas of research, particularly workflow management approaches, process modeling & enactment research, and software agents bear similarities to our work.

Most commercial workflow management systems like Staffware[1], FlowMark[2], or TeamWARE[3] focus on enactment in one company. The work process can be geographically distributed but the assumption is that the process is carried out within a company. They provide no support for a virtual software development corporation. Support for change management is not even sufficiently solved for single company solutions. With increasing numbers of VDSC and the focus of commercial WFMS will probably soon be adapted to the new requirements.

OzWeb [8] is a web-based system that supports multiple users, grouped together into collaborative teams. It is based on Oz that provides a rule-based process modeling language with a process engine using forward and backward chaining. It is based on an International Alliance metaphor that implies using dynamically definable treaties for building contracts (and handling access rights) between companies. But the main focus seems to be more on communication and interaction between companies than on supporting and coordinating a virtual software development process. The authors mention a *foundation* providing necessary coordination functionality, but the realization is not a goal of Oz. Hence they wanted to use a state/task server from a different university (ProcessWall) as foundation. In comparison, we focus on realization of coordination functionality and on management support, like resource allocation and time scheduling. This also leads to handling and triggering many kinds of project changes. Oz' successor project GRACE focus on the middleware layer which is not central to our research interest.

A project management system that does provide both planning and execution support is the Mesa/Vista Enterprise[4] tool, which is an environment for collaborative project execution and management. From the view on distribution, it supports large, geographically dispersed developer teams that belong to one company. It does not address the additional requirements from VSDC. A basic notification service exists, but needs to be installed manually. Complex plan changes are not supported.

Endeavors [3] supports distributed execution of (workflow) processes [9]. Endeavors provides support for dynamic process changes but most of it is "manual" work by its users. It is currently being extended to support World Wide Web (WWW) protocols. We could not identify support for negotiation or the different roles described above.

EPOS [12] is a software engineering environment with emphasis on process modeling, software configuration management and support for cooperative work. EPOS provides versioning and transaction management, controlled by an application-specific process model. It does not address issues of VSDCs.

The SPADE [1] project aims at defining and developing a software-engineering environment for software process modeling and enactment. Its process modeling language is based on a high-level Petri net formalism. SPADE addresses distribution and virtual corporation issues, but there are still requirements for VDSC that are not handled in this completed project.

Some aspect not mentioned in our first approach discussion of the negotiator agent are important and will be realized in a later implementation. It is useful that agents can subcontract not only a task but also a set of tasks to other agents. The advantage is that different partitions of the overall task set can be provided. Task sets reduce the messages send between the agents. Nevertheless, it gets harder for agents to calculate the effort for a set if alternative partitions are defined and subcontracts have to be taken into account.

We need to enhance bidding support in the future. It is useful to have information about calculation and bidding strategies of other agents.

Pre-selecting recipients of announcements would be helpful for both sender and receivers. It helps preventing message congestion, a well-known problem in multi-agent systems. In our first approach, an agent can advertises his capabilities to make other agents aware of it and all negotiation agents send offers to all agents they know. With increasing amount of agents, there will be increased messaging between the agents. This will lead to the need for pre-selection strategies to reduce the recipients of broadcasts. First, it is desirable that a pre-selection can be performed when tasks with special core competence will be announced. An expansion of our concept requires maintaining data about the core competence of other agent's companies. Making bad experience with a company is a reason for excluding it from the list of recipients. A possible approach is to maintain lists that hold information about this bad experience. A last point to mention is that subcontract loops should be avoided. If an agent gets a contract offer and it wants to outsource parts of the contract, it makes sense that the message will not go to the contractor.

Another point to mention concerns levels of commitment. In our first approach it is not possible to retreat from an existing contract. In reality, it is possible to retreat from a contract by paying a penalty for breach of contract. Our current negotiation support approach does not allow for a retreat from a contract. This is the same as if the penalty for a retreat is extremely high. Surely, it influences the decisions of the agent because a low penalty means a lower risk [13].

A contractee may ask for resources instead of money in a counterproposal. If, for example, a company has the core competence but not the resources needed to maintain the system, they may offer to use staff from the

[1] http://www.staffware.com/
[2] http://www.software.ibm.com/ad/flowmark
[3] http://www.teamware.com/
[4] http://mesasys.com/vistapm/

contractor. This staff can provide maintenance and support after development is finished.

Agents have no information about other agent's resources and tasks. In the future, our negotiation agents will try to collect this information. Although agents from other autonomous companies will not grant access to their data, agents from the same company or from tightly coupled companies may offer information about resources and tasks. This will lead to enhanced information to the human agent and so provide a better support.

Currently, our approach does not take into account time- decreasing/increasing payment. These functions can motivate people to accelerate their offers because a sooner response increases company earnings.

A special point we want to mention is the possibility that the supplier delivers products that do not meet an organization's needs [5]. A contractor can accept the delivered product, lower its standards, and help the supplier to meet the contractor's needs or choose another supplier. In the last case, a new negotiation phase will be started that has to take into account that time was wasted.

A next step will be the automation of the negotiation supporting agents.

Acknowledgements

This work is supported by NSERC, Nortel, the University of Calgary, and the DFG with several research grants. We would specifically thank Alexander Herzlinger and Sascha Mungenas for developing the replication components. Thanks also go to Barbara Dellen, Sigrid Goldmann, and Harald Holz for providing valuable input on MILOS.

References

[1] S. Bandinelli, E. Di Nitto, A. Fugetta. Supporting cooperation in the SPADE-1 Environment, *IEEE Transactions on Software Engineering*, vol. 22, no. 12, December 1996.

[2] F. Bendeck, S. Goldmann, H. Holz, B. Kötting. Coordinating Management Activities in Distributed Software Development Projects, *IEEE Post-Proceedings of the 7th Intl. Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*, Stanford (USA), 1998.

[3] G.A. Bolcer and R.N. Taylor. Endeavors: A Process System Integration Infrastructure, *Proceedings of the Fourth International Conference on the Software Process*, Brighton, England, December 1996.

[4] J.A. Byrne, R. Brandt, O. Port. The virtual corporation", Business Week, February 8, 1993.

[5] R. Bate, D. Kuhn, C. Wells. A systems engineering capability maturity model, Version 1.1, Software Engineering Institute, 1995.

[6] W. H. Davidow, M. S. Malone. The virtual corporation: Structuring and Revitalizing the Corporation for the 21st Century, New York 1992.

[7] J. Griese, J.Sieber. Virtualitaet bei Beratungs und Softewarehaeusern, *U.Winand, K. Nathusius (Hrsg.): Unternehmungsnetzwerke und Virtuelle Organisationen, 1998.*

[8] G.E. Kaiser, St.E. Dossick, W. Jiang, J.Jingshuang Yang and S.X. Ye. WWW-based Collaboration Environments with Distributed Tool Services, *World Wide Web Journal*, Baltzer Science Publishers, 1998.

[9] P.J. Kammer, G.A. Bolcer, R.N. Taylor, M.Bergman. Techniques for Supporting Dynamic and Adaptive Workflow. Submitted for publication, December, 1998.

[10] D.W. Karolak. Global Software development. Published by IEEE Computer Society, 1998.

[11] F. Maurer, B. Dellen: An Internet Based Software Process Management Environment. ICSE 98 workshop on "Software engineering over the Internet", *http://sern.cpsc.ucalgary.ca/~maurer/ICSE98WS/Submissions/Maurer/ICSE.html*

[12] M.N. Nguyen, A.I. Wang, R. Conradi. Total Software Process Model Evolution In EPOS. *4th ICSP*, Brigthon (UK), 1996.

[13] T. Sandholm, V. Lesser. Issues in automated Negotiation and Electronic Commerce: Extending the Contract Net Framework. Readings in Agents, 1998.

[14] R.G.Smith. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver, *IEEE Trans. On Computers* C-29 (12), 1980.

[15] J. Sieber. Virtuelle Unternehmen in der IT-Branche, die Wechselwirkung zwischen Internet-Nutzung, Organisation und Strategie, Dissertation, published in *Verlag Paul Haupt*, 1998.

[16] I.Z. Ben-Shaul, G.E. Kaiser. Federating Process-Centered Environments: the Oz Experience, *Journal of Automated Software Engineering*, 5(1), 1998.

[17] The Workflow Management Coalition. Workflow standard – Interoperability Abstract Specification, *Document Number WFMC-TC-1012*, 1