Approaching Support for Internet-based Negotiation on Software Projects

Boris Kötting[1], Frank Maurer[2]
[1]University of Kaiserslautern, Germany
[2]University of Calgary, Canada
koetting@informatik.uni-kl.de, maurer@cpsc.ucalgary.ca

Abstract

Negotiating a software development contract is a complex operation. One needs to make decisions about many different aspects of what information to provide to possible contractors, who potential collaborators can be and how one wants to negotiate with them. This paper deals with our approach for negotiation support for software development tasks. We created a virtual marketplace that supports contract negotiation between different companies as well as in between a company. In contrast to existing Web based marketplaces, we needed to develop concepts for dealing with complex structures like task hierarchies and also offer more negotiation possibilities.

1 INTRODUCTION

Optimal utilization of a company's own potential is crucial for its survival in the world markets. Many companies are not able to use their resources optimally and hence are not competitive over time. Some reasons therefore are:

- ?? The company lacks some competencies for performing special projects and hence cannot acquire these projects.
- ?? The execution of a large project would take to much time and hiring additional staff and other miscellaneous items can lead to a not desired strategic alignment of the company.
- ?? Employees and other resources have too much work in hot phases of development and not enough work when the project is in other phases.
- ?? Global operating companies should be able to perform projects globally, but they usually have tremendous problems in finding the right people for a given set of tasks and in organizing the collaboration.

Large software development projects often won't be performed by one company on its own but by a group of companies, either under control of one company leading the process or by a virtual corporation [6] of companies with equal rights. These points make it clear that it will be desirable for a company to contact other companies immediately for offering cooperation concerning the enactment of a project. This need for contact is desirable for both offerer and bidder for a service.

The need for communication about software development services motivates us for creating a virtual marketplace framework for negotiating software development tasks. Existing commercial marketplaces like ebay (http://www.ebay.com/) are only able to handle "atomic" items: an item is sold as a whole and bids for parts are not supported. There is also only one possible offering strategy, which describes how offerer and bidder interact with each other: the offerer puts an object with his minimal pricing expectation on the system. To enter a valid proposal a bidder need to provide a higher proposal than the existing one before the end date of the auction.

This approach is insufficient for complex software development tasks that often require being able to bid on parts of the overall task, for example on all Java-based implementation tasks. Research systems have a broader range of variability than the existing commercial systems, but there is no system that supports different negotiation strategies on complex and flexible structures. With [5,8] it is possible to create an agent that negotiates automatically, but the negotiation objects are just simple objects and the negotiation just concerns the price. [7] introduces a training environment for agents to improve the strategies the agents use in a competitive environment. But the simple object structure is also the handicap of this approach, Some systems [9] are able to negotiate multi-dimensional objects but the structure of the multi-dimensional object cannot be manipulated. There also is no possibility to use various offering types, like in [4], where different negotiation procedures are provided, based on a well-defined formal model. Nevertheless, this approach also doesn't provide negotiation on complex structures and sufficient flexibility.

In this paper we describe our approach for supporting contract negotiation of software development tasks. It provides various negotiation protocols on complex task structures and on parts of this structure concurrently. This support can help outsourcing, building virtual corporations or balancing the workload within a single company. The latter can be achieved by easily placing structured processes on the marketplace, so available employees can decide fast if they can perform these tasks. On the other

hand, it can help companies acquiring projects and concentrating on their core competencies. Here, the structure of the project will be much more complex and so is the negotiation about it. With our approach, project planners can offer projects of high complexity on a marketplace and afterwards negotiate inter-company or intra-company contracts about them or parts of them.

We assume that a software development project is described as a process hierarchy consisting of composite and atomic processes. Composite processes consist of subprocesses whereas atomic processes do not. All processes, consists of different information like a task schedule, necessary skills and incoming and outgoing products. With the latter it is also possible to define a product flow in between the complex tasks structure.

Our approach enhances the contract net protocol [3] that 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 [2], but all expansions that we found do not equip bidding agents with sufficient flexibility like bidding for parts of structures or changing the structure of composite processes.

The main focus of this paper lies on the concepts we designed for supporting the different roles in the project plan creation and negotiation process.

2 VMSDT Architecture

In 2000, we started developing a Virtual Marketplace System for software Development Tasks (VMSDT) that allows us to negotiate complex software tasks using a simple environment. VMSDT supports the auctioning of complex task structures and contract negotiation processes. It provides several auction protocols that can be used for offering and bidding processes for complex task structures. In our VMSDT it is possible to insert various information aspects concerning the task and additional information concerning the auction and negotiation protocol. VMSDT is integrated with the MILOS Workflow Management System that supports the globally distributed execution of the development process [1]. The architecture of the VMSDT can be seen in Fig.1.

A human user can use two different kinds of agents in the marketplace system, an offering agent and a bidding agent. With an offering agent a software development process can be placed on the marketplace. The offering agent assists the human user in various ways: the human user can use existing project plans. They just need to be transformed into valid XML-structures, which are already defined for offers. Updating data of running workflow engines is a more complex task but we also realized this for the MILOS-Workflow Engine.

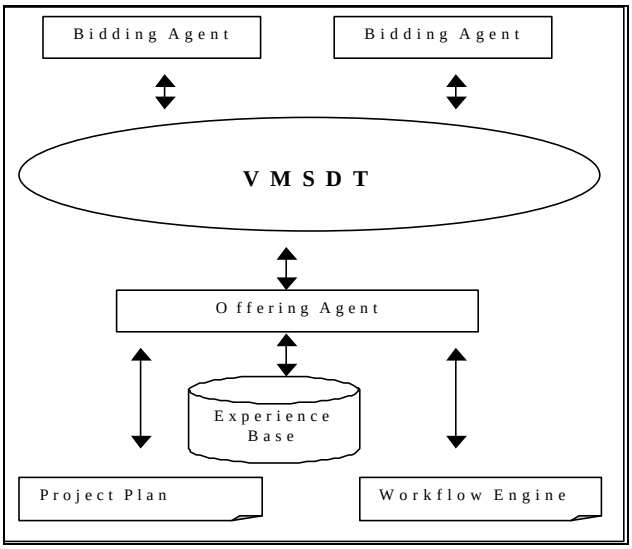

Figure 1: Marketplace System Architecture

On the other side, the bidding agent provides the orthogonal support: a user uses it to place bids for processes into the marketplace. A user that uses the bidding agent also can use the experience factory for improving her decision foundation. Additionally, the bidding agent can be configured by the human user to look for special processes in the marketplace, so the user will be freed of the burden to view every process entering the marketplace.

We put the negotiation logic into the objects that are put into and taken from the space. The planner has the ability to decide which information will be inserted into the object that will be written into the marketplace. The planner may also want to hide some information that is not relevant or sensitive to her company, like predecessor and successor tasks or internal reports.

Because of a clearly defined data structure and marketplace you have a formalized model for all agents, which eases communication. The data structure can be expanded to use additional information structures. Another advantage is that you can choose the recipient as exactly as you want: it is possible to address all agents, a group of agents or a specific agent in one of the virtual marketplaces of the marketplace system.

The task of an agent in the context of a virtual marketplace system is not only displaying data from the marketplace to the user but also pre-filtering of tasks that match her needs and goals as well as available resources of the company. The agent should also prepare knowledge from the experience base for additional information about former projects or other companies. This supports the decision of the concerned human user for or against the acceptance of a proposal, because it helps estimating effort or proving an evaluation of the reliability of an offerer.

2.1 Offering types

An offering type defines the protocol that is used to negotiate projects or parts of a project. It describes how offerer and bidder interact with each other and, especially, what freedom both sides have. We have realized different kinds of offering types (see also [4,10]):

?? Closed (yes/no) offerings: A bidding agent can only accept or reject an offer of a customer agent. No change of process parameters is possible.

?? English Auction: In case of an English auction, the offerer names the maximal amount she is willing to pay for a process, which automatically created the initial bid. During the auction, bidders can place bids with a lower price. When the closing time is reached or if no new bid is received within a predefined time interval, the auction is closed and the last bidder receives the contract for the process.

?? Sealed Auction: A planer offers a process on the marketplace with a fixed end time for bids. When this time is over, all bids received will be evaluated and one bid will be awarded as the winner. If a bidder writes more than one bid to the marketplace, only the last bid will be valid. This allows every bidder to react to changes in her environment that enable her to improve her bid. Variables in this protocol are time schedule and payment.

?? Negotiation: This is the most flexible protocol in our marketplace system. The offerer is able to mark every part of a process as variable to make it possible to negotiate about it. After she gets proposals from bidding agents, she can offer counterproposals and so on. Following this protocol, the two parties go a step forward towards a contract with every new proposal.

2.2 Element Structure

The element structure of offers and bids consists of information concerning three different parts:

?? Process specific information: we use attributes that reflect our experience with process modeling and are expanded by attributed from COTS-Tools. It is possible to define products and a product flow, so other agents can understand the existing contexts. Necessary skills for performing the task can be defined. These skills are a part of a skill ontology that is used for a specific marketplace. This helps matching the right tasks to the right companies. Furthermore, you can insert scheduling information, i.e. planned start and finish times and estimated effort. Finally, you can insert the description of the task and its goal.

?? Contract specific information: the contract part was developed based on a literature survey [10], some contracts signed by the university and some contracts from industry. We will not discuss the intrinsic difficulties of working law here because it is beyond the scope of the paper. In our system, a contract consists of elements like requirements, milestones, equipment, warranty, communication, cost structures and miscellaneous. There are some contracts in our contract experience base that can be loaded at runtime to abbreviate the creation of new contracts for every new task.

?? Negotiation specific information: an important attribute of this part of the element structure of offers and bids are the payment structure and conditions. VMSDT supports the user in calculation of the payment. You can calculate it based on its subtasks but also based on the estimated effort of person days. Of course, you can also simply insert a price value. It is possible to send the object to a specific set of users, so that the information overload of participants in the marketplace can be reduced and agents that are not welcome, like untrustworthy companies or competitors that want to gain information about the company's projects, can be left out. There are user groups that are maintained by the administrator but every agent can additionally customize own groups of addressees for an exact match of his necessities and for making the addressing procedure more comfortable. Finally, this part of the object contains one of the offering types that were already discussed in this chapter.

3 Complex Project Structures

While negotiation on simple objects with only one offering type can be realized quickly, the step to complex project structures and different offering types is difficult. One of the most important reasons is the possibility to negotiate on different, none disjunctive parts of a project concurrently. In the following we distinguish between atomic and composite processes, as explained in Section 1. The ability to offer complex structures like composite processes brings up a several points to consider.

3.1 Partial Offering

A partial offering is only possible with complex or large tasks, not with atomic tasks. A bidding agent receives a contract offer but can only handle some of the task that need to be performed. So he puts a "partial bid" object on the virtual marketplace, where she can name the components she wants to perform and the payment she wants to receive. This case leads to one of the offering types described above. If the offering agent wants to accept the partial offer of the bidding agent but not for the price requested, she can put a closed offering on the marketplace. This offer is addressed to the bidder - other agents will ignore this object when it is addressed to a special agent - with a lower payment.

To illustrate different aspects, we look at a small project plan example (Fig.2). Atomic processes in this

example start with an “A” (e.g. A1) composite processes with a “C” (e.g. C1).

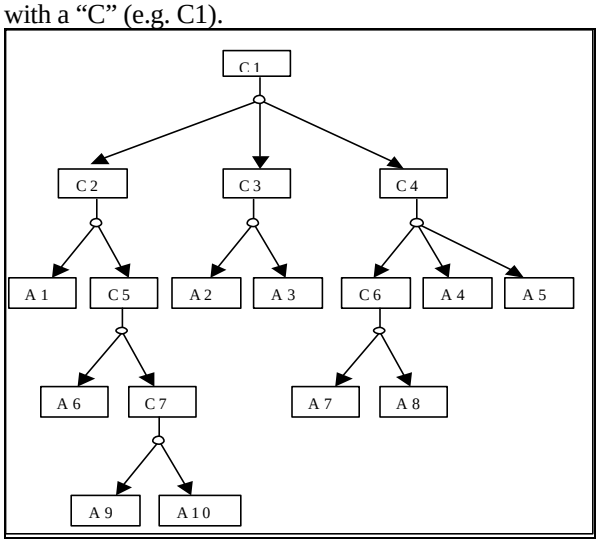

Figure 2: Project Plan Example

Let’s assume that the planner decides to offer the process C1 on the marketplace. He has a few restrictions on the structure:

?? There are indivisible composite processes in the project: the composite process C5 cannot be performed by more than one company so it should be forbidden to make a bid for the processes A6, A9, A10 or C7. In our system, the planner has the possibility to mark the process C5 as a *Union*. The processes in this union then have the state *Part of Union*. This state implies that only bids are possible for the whole union.

?? There are composite processes that must not be performed by a single company: the composite process C2 should not be performed by only one company. An example for that could be quality assurance of a software product: the same company should not perform implementation and testing. In VMSDT the planner can mark the task C2 as *Only Subprocess Contracts Possible* (OSCP).

There are subprocesses that contain company sensitive information: the process C6 and its subprocesses contain information that is sensitive but also contain necessary information for performing the tasks A4 and A5. The planner has the possibility to mark the process C6 (and its subprocesses A7 and A8) as *Excluded* from the offer, meaning that no bid for these processes can be performed and no information is accessible by other parties. It is obvious, that processes have a more complex state in this scenario. When dealing with atomic tasks only, it is sufficient to model the state of a task with a single attribute *accepted* which is set to true or false, so you receive two possible states for atomic tasks. When working with composite processes it is necessary to extend the set of possible states of a process. New states are derived from the situations described above:

?? If a process is in the space for information purposes and it is not possible to negotiate about it, its state attribute *excluded* is set.

?? If negotiation is allowed for the process and not for the subprocesses, the state attribute *union* is set.

?? If it is not possible to negotiate about a process because it is part of a union, its state attribute *part of union* is set.

?? If it is not possible to negotiate about a process because it is only possible to negotiate about its subprocesses the state attribute *oscp* is set.

A process can have several state attributes set at the same time; e.g. it is part of a union and excluded.

3.2 Inheritance and propagation of attributes

When negotiating composite tasks, it is necessary to decide which attributes should be inherited by subtasks:

?? Schedule propagation: this can be seen as an aid for the planner, not as a strict order to follow. When a planner defines the schedule for *all* atomic processes, she is able to calculate the schedule information for the composite process containing these subprocesses. Here we do not follow critical path planning, but only look for the earliest start date and the latest end date of all atomic processes. This calculation is not mandatory because the duration of the composite process can vary from the single atomic process duration; e.g. the schedule for the composite process can be shorter because there is a smaller communication overhead within one company. When a company performs all subprocesses there is a smaller communication overhead, so the composite process can be executed faster. To make it more attractive for a bidder to bid for the composite process instead of bidding for the atomic processes (because of a timely advantage), the net benefit for the composite task usually is higher.

?? Price propagation: analogously to the schedule propagation, a planner is able to determine the price of a composite process based on the prices of its atomic subprocesses. As with the schedule it is also not mandatory, so the planner has the freedom to decide for a different price. We include a tool that supports the human user by providing the possibility to combine the calculation of employees-days or only days with the time schedule for the process. I.e. the user can calculate the price by using a fixed amount of money for a day and the calculator offers him an estimate for the project price.

?? Skill propagation: skills required for executing atomic processes will be propagated to the composite processes. To perform such a process, the union of all necessary skills for all subprocesses is necessary.

?? Contract Inheritance: this is a desirable feature because

it would save the human user a lot of work if it wouldn't be necessary to define a new contract for every subprocess. Unfortunately, this is very complex to handle. A lot of assumptions have to be made: a contract for a subprocess can be completely different to a contract of the same process as part of a composite process. So we use the strategy that every process has its own contract fields. These are completely independent of its parent process. But if a composite task has been assigned to a company, the contracts for the subprocesses are obsolete. In terms of object orientation: the contracts of the subprocesses are overridden.

?? Inheritance of negotiation strategy: allowing multiple negotiations on different -not necessarily disjunctive- subparts of a project leads to conflicts concerning the negotiation semantic. Every element that is placed on the marketplace is required for finishing the project. When a contract for one of the subtasks will be reached, it is not possible to negotiate about the whole parent tasks anymore because some of its subtasks are already assigned to a contractor. Assume that an English auction is started for a composite process and an English auction is also started for all of its subprocesses. If a valid bid for the composite task is in the marketplace and concurrently an agreement for a subprocess will be reached, then the bid for the composite task must be invalidated. This needs to be done because it was a bid for the task and all of its negotiable subtasks, but one of the subtasks is already accepted by another party. As mentioned above, all bids in the marketplace are mandatory in the marketplace system, so allowing English auction for composite processes and its subprocesses concurrently leads us to an inconsistency we cannot accept. Therefore, we made a distinction between two kinds of strategies, keeping in mind that it is possible to avoid negotiation about the subtasks by creating a union. We decided to allow English auctions and Closed Auctions only for atomic processes and unions. The arguments for the Closed Auction are similar to the arguments for the English auction. If one of these strategies is selected for a composite process, it will automatically be made a Union. For Sealed-Bid Auction and Negotiation, we allow strategy inheritance. This can be an advantage for the offering agent, e.g. for a Sealed Bid Auction. Here, the offering agent can receive bids for tasks as well as bids for all of the subtasks and finally decide if it is better to give the whole task structure in one hand or not. This is not a violation of the underlying semantic, because it reflects economic reality on non-virtual marketplaces.

?? Inheritance of process state attributes: in general, process state attributes will be inherited and can be more restrictive downwards. A subprocess of a *blank* process is by default also *blank*. Subtasks of a task that is *part of a union*, are also parts of a union. Subtasks of *excluded* tasks will also be *excluded*. There is an exception for subtasks of a *union* task. They have the state *part of union* instead of the state *union*.

3.3 Process State Change

Changing process states can happen frequently, so one need to consider the effects of a change.

Atomic processes can change their state from *not accepted* to *accepted*, *part of union* or *excluded*. The latter is not possible if the atomic process is the root process: it wouldn't make any sense to put a sole process on the marketplace, which is not negotiable. Atomic processes can not have the state *OSCP* because they have no subprocesses. A change into the state *part of union* can only happen indirectly if a parent state changes to *union*. Changes to *excluded*, *accepted* and *blank* can happen both directly and indirectly.

For composite processes there is a rule of thumb: when a composite task changes its state, its subtasks will also get the same state.

There are some exceptions to this rule: if the parent gets the state *union* the subprocesses change their state to *part of union*. A composite task can change its state to *OSCP*, which has no effect on the subtasks. If a complex process changes its state from excluded to one of the states blank, union or OSCP, the subprocesses still are excluded. When the state of a task changes to *accepted*, its subtasks are only *accepted* if they are not *excluded*.

The discussion of complex project structures is still in progress. It seems that there are still a lot of aspects that we need to clarify in the future.

4 State of the implementation

We implement the virtual marketplace system in Java. We build our implementation upon TSpaces™ technology (http://www.almaden.ibm.com/cs/TSpaces/index.html), which is also implemented in Java. It features persistence, object matching and notification. Our prototype consists of more than five hundred newly created classes. On server side we have user interfaces for starting and maintaining the marketplace system. On client-side we have several interfaces for offering and bidding agent, e.g. on the offerer side an interface for creating a complex task structure and on the bidder side a bidding assistant for creating a bid for an offer.

In VMSDT, there is no direct interaction between agents. They communicate via objects in the marketplace. So, if an agent broadcasts an offer to other agents (as described in the Contract Net Protocol), he writes it to the virtual marketplace, addressed to everybody who may be interested. Both agents need to support different interfaces: to the human agent (graphical user interface), to the virtual marketplace system and to the experience base. The virtual

marketplace system itself offers necessary negotiation protocols and data structures. It uses a predefined interface to TSpaces™, a tuplespace technology that is used as the underlying database system for our VMSDT.

The realization of the concepts will be finished by summer 2001. The different negotiation protocols are already tested for atomic processes and realization for composite processes is in progress. For atomic processes the workflow integration with the research prototype MILOS was successfully implemented on the offerer side.

We added the possibility to store and restore the project structure via XML-Files. This has several advantages: first, we have a clearly defined data model of our project structures. Second, we easily can transform the structures to XML-structures of existing project planning tool and vice versa and third, the user can save his work during creation. This is necessary, because the structures that are created with the tool can become very complex.

We put a lot of effort in supporting the user in creating consistent tasks. For this reason, we developed a consistency manager that informs the user about inconsistencies in his task structure. This happens for example triggered by wrong user input or when the user clicks on so-called *inconsistency markers*. These markers are visible on locations where the consistency manager found inconsistencies.

5 Future Work

In the near future, our work concentrates on finishing the implementation of the concepts we described in this paper.

Evaluating the marketplace system in some companies is our long-term goal. For now, we decide for pragmatic reasons to validate the marketplace system by creating real-world scenarios and running simulations on the marketplace system. This will allow us to determine the overall throughput of projects for a given set of resources and companies.

We also want to extend the contract experience base, so that the user has a bigger and better choice for her projects. We plan to distinguish between different contract categories like Framework-Contracts, Work-Contracts and Service Contracts.

There will be work on automating negotiation agents for atomic tasks. This can help both the planner and the employee. An employee can start her bidding agent, defining which kind of tasks she is willing to do. Her automated agent tries to negotiate with other agents to get tasks for her automatically. A planer can simply place the tasks on the marketplace without assigning them to specific agents, saving a lot of time. An automatic negotiation agent will be started for every available employee. The agent is pre-configured and bids for tasks matching his profile.

If new offering types are necessary, we will enhance our marketplace system with new negotiation strategies, like the Dutch auction.

Acknowledgements

The Deutsche Forschungsgemeinschaft DFG and the Bundesministerium für Bildung und Forschung BMBF support this work with several research grants. Thanks go also to Boris Schadt for the valuable discussions concerning support for complex task negotiation. Additional thanks go to Jörg Hohwiller for implementing large parts of the first prototype version and also for valuable discussions about different negotiation strategies.

References

1. F. Maurer, B. Dellen, F. Bendeck, S. Goldmann, H. Holz, B. Kötting and M. Schaaf: Merging Project Planning and Web-Enabled Dynamic Workflow Technologies, IEEE Internet Computing, Vol. 4, No. 3, May/June 2000.

2. Sandholm, V. Lesser. Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Framework. Readings in Agents, 1998.

3. Reid G. Smith. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem solver. IEEE Transactions on Computers, Vol.12, 1980.

4. G. Cass, H. Lee, B. Staudt Lerner, L.J. Osterweil, Formally Defining Coordination Process to support Contract Negotiations, University of Massachusetts Computer Science Technial Report UM-CS-1999-039, 1999.

5. A. Chavez, P. Maes. Kasbah: An agent marketplace for buying and selling goods. Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, London, 1996.

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

7. E. Gimenez-Funez, L. Godo, J.A. Rodriguez-Aguilar. Designing Bidding Strategies for Trading Agents in Electronic Auctions. Proceedings of International Conference on Multi Agent Systems (ICMAS '98), 1998.

8. C. Preist. Economic Agents for automated Trading. Hayzelden, Bogham (Eds.): Software Agents for Future Communication Services, Springer 1999.

9. Karolak. Global Software Development, 1999.

10. N.Vulkan, N.Jennings. Efficient Mechanisms for the Supply of Services in Multi-Agent Environments. Decision Support Systems, 28, 2000 (5-19) (April, 2000.)