Appears in *Special Issue of the Group Decision and Negotiation Journal on "e-negotiations", 2003*

E-Alliance: a Negotiation Infrastructure for Virtual Alliances

Stefania Castellani, Jean Marc Andreoli
Xerox Research Centre Europe, 38240 Meylan, France
*(*Stefania.Castellani,Jean-Marc.Andreoli@xrce.xerox.com*)*

Mihnea Bratu, Olivier Boissier
ENS Mines de Saint-Etienne, 42023 Saint-Etienne, France
*(*Mihnea.Bratu,Olivier.Boissier@emse.fr*)*

Ilham Alloui, Karim Megzari
LISTIC, ESIA, BP.806 - 74016 Annecy, France
*(*Ilham.Alloui,Karim.Megzari@univ-savoie.fr*)*

Abstract. In this paper we provide an overview of E-Alliance, a software infrastructure we are developing to support negotiation activities in concurrent inter-organisational alliances. Our baseline is to offer a collaboration framework which fully preserves the autonomy of organisations grouped in an alliance, while enabling concurrency of their activities, flexibility of their negotiations and dynamic evolution of their environment. We propose to support negotiation between the partners within such alliances by combining different technologies, such as software engineering techniques, middleware-level coordination facilities and multi-agent systems support. We present our approach in the context of a sample scenario of an alliance where partners are printshops capable of (out/in)sourcing print jobs among them to better accomplish their customers' requests.

Keywords: negotiation, middleware, virtual enterprises, multi-agent systems, interaction protocol

1. Introduction

The advent of the Internet has led to the flourishing development of various forms of virtual organisations which try to exploit the facilities of the network to collaboratively perform tasks that span across the boundaries of the member organisations. What was previously performed by ad-hoc means within each organisation could now be realised by standardised means across organisations. However, this transition raised a number of issues which have not yet been fully resolved. In particular, while the pure communication issues have been satisfactorily dealt with through agreed standards, no such consensus exists on how communication should be used to achieve higher level goals.

Typically, workflow systems have developed generic models of process enactment, that could span across organisations, using the shared communication infrastructure. Workflow solutions have proved successful whenever the "rule of the game" of the collaboration among the members of a virtual organisation is clear and easy to formalise in some generic model. Everything works fine as long as the rule of the game need not be modified nor adapted, but actual workflow practice demonstrates that exceptions occur, where the players have to change the rule to make it work. Workflow systems have developed various solutions to the problem of exception processing, allowing some form of dynamic modification of the rule of the game. In fact, these solutions usually amount to setting more rules, saying what should be done when the previous rules cannot or should not be applied. Such an accumulation of rules may be confusing and has been a major hindrance for the adoption of raw workflow solutions in the context of virtual enterprises. At the other end of the spectrum, tools for computer supported collaborative work have been developed to provide support for very dynamic and loosely structured collaborations among processes. In the context of virtual enterprises, this kind of support is not satisfactory: B2B interactions need to be explicitly constrained by general rules of behavior agreed upon by the participants.

We therefore seek to achieve an intermediate solution to provide support to the collaborations within an alliance of organisations and we propose *negotiation* as a fundamental mechanism for these collaborations. We are developing the E-Alliance infrastructure, based on this approach, which allows an organisation to dynamically join or leave an alliance and make autonomous decisions to progress in its collaborations. Negotiation-based collaborations may occur asynchronously following very different patterns, but of course do not preclude prior agreement between the alliance members as to how negotiation should execute. An *e-alliance* is more structured than a shared dataspace for collaborative work, but less structured than a workflow assigning precise roles to participant organisations and tightly scheduling their interactions.

In this paper we describe the current status of E-Alliance, showing how organisations participate to and control the status of the negotiations and how the alliance life-cycle is managed. To illustrate our approach, we use a sample B2B scenario (Sec. 2) where autonomous printshops form an alliance to better accomplish their customers' requests. Sec. 3 describes goals and design requirements for the E-Alliance infrastructure. Sec. 4 provides an overview of the E-Alliance approach. Sec. 5 to 8 describe the support we are developing to fulfill the requirements. Sec. 9 shows an example in the context of a printshops alliance. Finally, Sec. 10 discusses related work.

2. A Scenario

We consider here a scenario (Andreoli et al., 2000) of collaborations within an alliance of distributed autonomous printshops. The alliance is a dynamic entity where new printshops may join or leave. A printshop manager interested in joining an alliance fills in an adhesion contract with information on his printshop competencies and preferences. If the alliance committee accepts it as a new partner, the new member commits to respecting the rules of the alliance and the adhesion contract and introduces itself to the other partners.

Each printshop autonomously manages its contracts, schedules etc. When a print request reaches a printshop, the manager analyses it to understand if it can be accepted, taking into account job schedules and resources availability. If the manager accepts the print request, he may decide to perform the job locally or to (partially) outsource it, given the printshop resource availability and technical capabilities. If the manager decides to outsource a job, he starts a negotiation within the alliance with selected participants. The manager may split the job into slots, notifying the partners about the outsourcing requests for the different slots. If the negotiation results in an agreement, a contract is settled between the outsourcer and the insourcer printshops, which defines an inter-organisational workflow enacting the business process fulfilling the outsourced jobs and a set of obligation relations among participants.

3. E-Alliance Requirements and Goals

The printshops alliance scenario shows a typical example of the e-alliances targeted by E-Alliance: virtual alliances where partner organisations may *a priori* be in competition with each other, but may want to cooperate in order to be globally more responsive to market demand. A lot of flexibility and coordination among the partners is needed to publish selected information, reach agreements on how and when to accomplish customers requests, execute and monitor contracts, handling changes. E-Alliance main goal is to provide a software support for inter-organisational alliances enabling:

1. management of an alliance's life-cycle, including services for information publishing, partners authentication, joining/leaving the alliance;
2. collaborative activities among alliance partners, through services enabling the partners to negotiate, execute and monitor contracts.

E-Alliance should flexibly support negotiation activities in the alliance respecting the autonomy of the partners without statically attaching each negotiation participant a role according to a strict protocol. Also, the mechanisms supporting such collaborations should be generic enough to adapt to any B2B

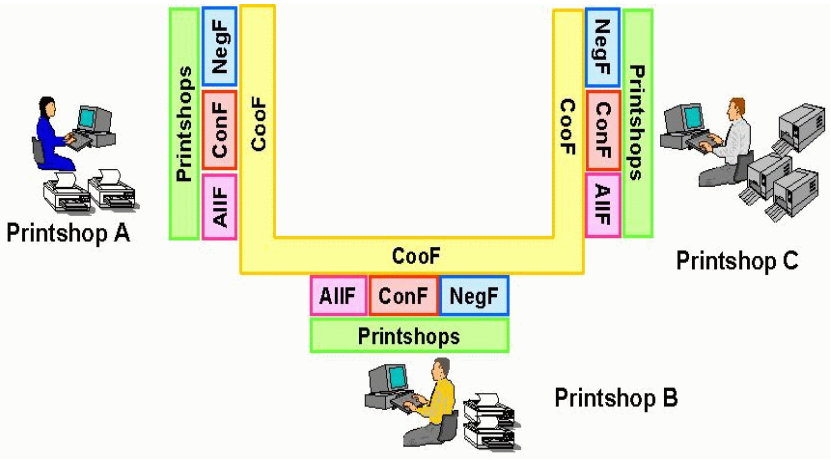

Figure 1. E-Alliance software infrastructure.

context. Moreover, E-Alliance should help the partners to augment their efficiency and ability to react to unforeseen situations, thus improving their market competitiveness. These issues have not been completely explored yet, although a lot of work has been done on other aspects (e.g. how to define payment mechanisms.) We focus instead on how to: (1) represent decentralized organisations; (2) model the coordination of different concurrent interactions; (3) formalise negotiations; (4) deploy and maintain an alliance during its life-cycle; and (5) create and administrate contracts. Next section describes how the E-Alliance approach takes into account the discussed requirements.

4. E-Alliance Approach

The E-Alliance infrastructure proposes a multi-level architecture for providing services to assist alliance partners along their collaborative concurrent activities taking into account the aspects discussed in Sec. 3. The infrastructure (Fig. 1) is organised in three layers. A first, application dedicated, layer specializes the generic mechanisms provided by the other two layers according to the specific domain, e.g. the printing domain. A second layer is dedicated to the support of job insourcing/outsourcing within an alliance and comprises three facilities: *AllF* (alliance life-cycle management), *ConF* (contract management), and *NegF* (negotiation). The third, middleware and coordination, layer (*CooF*) offers generic mechanisms to enact negotiations in a distributed environment. The *CooF* is shared across the partner sites, while the two other layers are replicated on each partner site, enabling a decentralized negotiation and preserving the autonomy of the partners.

Negotiation is part of a more global sub-contracting process aiming at defining contracts for jobs outsourcing and insourcing. Negotiation is carried on as if the job under negotiation was on a table, (partially) visible to each participant. All the alternatives in the negotiation can progress in parallel, asynchronously as long as no commitment is taken; each participant reacts to changes in its environment and to the evolution of propositions. Alternatives act as constraints that progressively reduce the range of potential agreements, and can be reactivated later during the process of contract execution in case of failure. Thus, negotiation can be considered as a Distributed Constraint Satisfaction Problem (Bui and Kowalczyk, 2001). The "distribution" part deals with constraint propagation between nodes, coupled with consensus making across multiple nodes, while the "satisfaction" part deals with constraint-based reasoning and strategic reasoning at each node. In our approach, this decomposition corresponds to the role played by the *CooF* and *NegF*. The *CooF* provides an infrastructure with generic mechanisms for consensus and constraint propagations: broadcast, alternatives synchronisation, transactions. A *NegF* offers a set of services taking place between a manager and the *CooF*, automatically processing parts of the negotiation process.

5. Alliance Facility

The goal of the *AllF* facility is to support an alliance life-cycle including: new members subscriptions and members departures, modifications of adhesion contracts, of members preferences, of the global rules of the alliance. This addresses two major issues: (1) what kind of software architectures cope at once with autonomy, openness and evolution requirements of alliances; and (2) what processes to put in place in order to specify, enact, deploy such alliances. In order to maintain the global state of the alliance and to provide managers with the appropriate information, an *AllF* supervises the activities of the *NegF* and *ConF* to check whether the rules of the alliance are respected or not. It also gathers information in order to build a global history of the system. If an event in the life of the alliance has an impact on ongoing negotiations and contracts, the *AllF* interacts with the concerned facility in order to maintain the global coherence of the system. The information manipulated within the alliance include *global* information, e.g. adhesion contracts, and partner *local* information, e.g. its representation of the others. An adhesion contract expresses the engagements between the alliance and a member, e.g. services the member will provide.

E-Alliance provides a software environment offering the users means for dynamically adding/retracting/replacing software components, without interrupting the system execution. The underlying approach relies on modelling an alliance life-cycle using the Zeta (Alloui and Oquendo, 2001) architecture

description language and generating an executable code from the description into a target implementation environment called ProcessWeb (PML, 1996). The resulting software environment allows the *AllF* to communicate with both *NegF* and *ConF* facilities, e.g. to provide the *ConF* with rules to apply to a contract or to record in the history a new contract or negotiation.

6. Contract Facility

The *ConF* facility of a partner of the alliance manages the execution of the contracts in which that partner is involved in. The management of a contract is organised around three main steps: (1) *creation*; (2) *execution*; (3) *closing*.

During the creation, the *ConF* of the principal contractant defines a contract from the terms of the agreement reached during the negotiation by its *NegF* and the *NegFs* of the other participants in the negotiation. A lot of B2B contract models have been proposed in the literature (Grefen and Angelov, 2001). In E-Alliance a contract is composed of a business process, fulfilling the negotiation agreements, and of a normative and policy structure, ruling the participants behavior. Using the $\mathcal{M}$OISE$^+$ model (Hubner et al., 2002), we define the structure of a contract as a set of *roles* linked with each other with authority and communication *links*. This structural schema sets the authority structure that governs the contract. The responsibilities of the *ConF* of each contract participant are defined by linking the *roles* it can play with the part of the business process that it has to execute. These links are expressed as obligations or permissions, and are qualified by penalties in case a participant cannot fulfill a task it is responsible for.

Executing a contract consists of the distributed execution and enactment of an inter-organisational workflow between the participants, and it is supported by the *CooF*. Different events may stop the execution of a contract and imply modifications of it. These events may be communicated by the *AllF* as a consequence of a change in the alliance itself, but they may also result of impredictable changes. The *ConF* will interact with the *NegF*, if a new negotiation is needed, and with the *AllF* to make it aware of the penalties for the participants.

7. Negotiation Facility

The *NegF* agent of a partner manages the negotiations the partner is involved in. Fig. 2 shows the architecture of a *NegF*, which assists its manager at a global level (negotiations on *different jobs*) and at a specific level (negotiation on the *same job* with different participants) by coordinating itself with the *NegF* of the other partners through the *CooF*. A negotiation is organised in

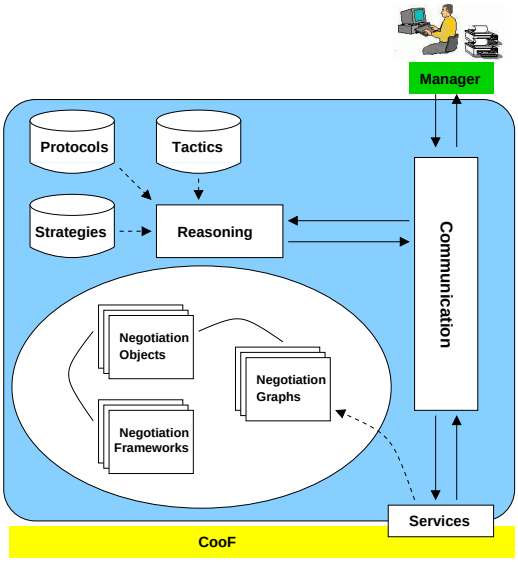

Figure 2. The architecture of the *NegF* facility.

three main steps: initialization; refinement of the job under negotiation; and closing. The initialization step allows to define what has to be negotiated (*Negotiation Object*) and how (*Negotiation Framework*). A selection of negotiation participants can be made using history on passed negotiation, available locally or provided by the *AllF*. Following the approach in (Vercouter, 2000), each participant has its own representation of the other participants and uses it to build a network of dependence relations (Sichman et al., 1994). In the refinement step, which relies on a set of speech acts (Carron et al., 1999), participants exchange proposals on the negotiation object trying to satisfy their constraints.

The manager may participate in the definition and evolution of negotiation frameworks and objects. Decisions are taken by the manager, assisted by his *NegF* agent. Decision functions operate in the "Reasoning" box (Fig. 2), totally or partially automating the negotiation. For each negotiation, a *NegF* manages, one or more negotiation objects, one framework and the negotiation status, detailed in Sec. 8. A *Negotiation Object* has the following structure:

⟨ON⟩ ::= ⟨SON⟩| '('⟨name⟩⟨ON⟩* :dependencies ⟨DDF⟩')'
⟨SON⟩ ::= '('⟨name⟩ :type ⟨TDF⟩ :candidates ⟨CDF⟩ :job ⟨JDF⟩ :dependencies ⟨DDF⟩')'
⟨CDF⟩::= '('⟨CDF⟩* [⟨Relation⟩]')' | ⟨Name⟩| '?' | '?*'
⟨Relation⟩::= >> | <<
⟨JDF⟩::= '("('⟨Issue⟩ (⟨Constraint⟩) * [⟨Preference⟩]')' * ')'* [⟨Relation⟩]
⟨Issue⟩ ::= ⟨IssueName⟩ (⟨value⟩| '?')
⟨Constraint⟩ ::= '(' (⟨c_Exp⟩|⟨Constraint⟩)* [(and | or | xor)]')'
⟨c_Exp⟩ ::= ⟨IssueName⟩⟨Operator⟩⟨value⟩
⟨Operator⟩ ::= < | <= | > | >= | == |/ = | enum

⟨DDF⟩ ::= '(' ⟨Exp⟩* ')'
⟨Exp⟩ ::= '(' ⟨val⟩* [⟨d_Operator⟩]')'
⟨val⟩ ::= ⟨IssueName⟩|⟨Integer⟩|⟨Exp⟩
⟨d_Operator⟩ ::= + | - | × | / | = | if | < | <= | > | >= | == |/ =

A negotiation object can be composed of several negotiation objects, which can be interdependent thus expressing interdependencies among concurrent negotiations. Otherwise, it is a simple negotiation object (SON) with a unique type defined in the domain ontology of the considered application (TDF). In the print tasks domain, a type could be black and white printing. The manager can specify one (?) or more (?*) negotiation candidates, group and order them so that during the negotiation, an attempt is made in turn for each set of partners. The job under negotiation is specified as a set of (possibly interdependent) issues derived from the domain ontology of the considered application. For example, in the print tasks domain an issue can be the cost. Constraints are used to specify possible values for the issues, e.g. (job.cost<=12). A preference can be expressed as a utility function for each issue in order to calculate the value of an offer (Keeny and Raiffa, 1976). Relations allow to sort the sets of issues which will be considered one after the other during the negotiation (multi-stage protocol). Dependencies between issues can be expressed through algebric and relational operators. For a complex object, the negotiation has to be carried on satisfying the constraints specified in the complex object and the ones defined in the component objects. For example, if ON0 is an object composed of ON1 and ON2, the dependency (ON1.cost (ON2.cost 2 *) >=) in ON0 states that the cost for the job in ON1 must be greater or equal to the cost for the job in ON2 times two.

Negotiation Frameworks gather requirements of managers on negotiations, formalising plans for the interaction process and the degrees of autonomy in decisions and actions of the *NegF*. A negotiation object has a unique negotiation framework, but a negotiation framework can cover several negotiation objects. A negotiation framework has the following structure:

⟨CN⟩ ::= '('Context :duration ⟨duration⟩ :messages ⟨integer⟩ :candidates ⟨integer⟩
:contractants ⟨integer⟩ :strategies ⟨strategy⟩*
:tactics ⟨tactic⟩* :protocols '(':manager ⟨name⟩ :negf ⟨name⟩')'

A manager can specify some global parameters: duration; maximum number of messages to be exchanged; maximum number of candidates to be considered in the negotiation and involved in the contract (contractants); tactics; protocols for the *NegF* interactions with the manager and with the other *NegFs*. Differing from (Faratin, 2000), where tactics are defined for managing the negotiation, here tactics define constraints on the negotiation process. They do not bear directly on a proposal, but on the entire set of exchanged proposals specifying constraints on the coordination of the propagation of alternatives

to other negotiation participants. For example, a tactic may state that a job has to be outsourced as a block, another one that it has to be split in several slots. Executing a tactic corresponds to activating a combination of *services*, implemented above the *CooF*, producing a coordinated modification of alternatives within the current negotiation. Each service manages a local view of the global negotiation, translating negotiation decisions to modifications on the set of the visible alternatives on the job under negotiation using primitives of the *CooF*. A number of services connect the *NegF* to the *CooF*, including:

- `outsrc` (resp. `insrc`), for outsourcing (resp. insourcing) jobs by exchanging proposals among participants known from the beginning;
- `split`, for propagating constraints among several slots, negotiated in parallel and issued from the split of a single job;

We plan to provide each *NegF* with a library of protocols, but currently we use a modified version of the Iterated Contract Net Interaction Protocol (ICNIP) (FIPA) which allows to manage a multi-stage negotiation according to an ordered set of issues (or of candidates) as defined in the negotiation object.

8. Negotiation Middleware

The *CooF* is the negotiation oriented coordination middleware that supports the different processes provided by the facilities in the second layer of the E-Alliance infrastructure. It is an extension of the CLF middleware (Andreoli et al., 1999) aiming at enriching its negotiation support capabilities (Andreoli and Castellani, 2001). In CLF all the components are viewed as resource managers. Resources can be "tangible" elements, e.g. a printer, as well as more virtual entities, e.g. a print task. CLF components make visible their resources through interfaces that define abstract services through which operations on resources are made possible. The interaction with a CLF component through one service of its interface follows a specific protocol, defined by eight "interaction verbs", similar to speech acts, which have a meaning in terms of resource manipulations (discovery, selection, insertion and destruction). The coordination of these components, considered as resource managers, is expressed by means of high-level rule-based scripts hiding the communication protocol and directly expressing the desired resources manipulations. Specific CLF components, called *coordinators*, translate scripts into invocations of the protocol on different components, realising the abstract resource manipulation prescribed by the script. Thus, the coordinators can be considered as generic clients of the middleware platform and the client side of a CLF application can be expressed as a set of scripts. As an example, a printshop in the alliance scenario could be represented by a CLF component offering services

for outsourcing/insourcing jobs (Andreoli et al., 2000). The resources held by this component are decisions to outsource or insource a job.

A simple outsourcing mechanism is modeled by the following CLF rule:

```
outsrc(job) @ partner(d,job) @ insrc(d,job,offer) @
accept(d,job,offer)
<>- transfer(d,job,offer)
```

To be triggered, the rule requires the following resources together: a job to outsource (`outsrc`), provided by a component of type printshop; a link between the job and a potential insourcer partner (`partner`), provided by a component of type yellow pages; an offer made by that partner for the job (`insrc`); an acceptance of that offer (`accept`) generated by the outsourcer partner. All these resources are searched in their respective components using the search capabilities of the CLF protocol. The search phase consists of the asynchronous construction of a "search tree" encoding all the possible combinations of resources of the above type. When a branch in the tree is complete, ie. a resource has been assigned to each token in the left-hand side of the rule, the enactment capabilities of the CLF protocol allow to atomically consume the corresponding resources and notify success by insertion of the resources specified on the right-hand side of the rule (`transfer` in the example).

On the server side, CLF offers a rich library of classes for building servers able to answer the protocol. Some prototypical servers, e.g. databases, have been encapsulated as ready to use CLF components and used in several prototype applications for supporting "workflow". Building upon this experience, we are elaborating CLF-based support tools for contracts execution and monitoring in E-Alliance, transparently offering transactional and notification capabilities. The resources involved here represent the state of progress of the contracts execution and the different actions performed by the contractors.

From CLF to a Negotiation Middleware The CLF protocol allows to perform multiple dependent searches for resources held by several components, but it enables only a "uni-directional" propagation of the information through the involved components. Indeed, each response sent by a server in the search phase must be a "complete" specification of the *actual* resource (e.g. a given print job) to be used in the enactment phase if and when it is performed. The server cannot return "partial" answers describing a set of *potential* resources, then letting the client refine that set in order to converge towards the resource to be used in the enactment, if any. Thus, in our example, the search for the `insrc` resource will create branches in the search tree, each containing a complete description of the insourcer's offer, which the outsourcer may only accept or reject as such, by proposing or not the resource `accept`. The *CooF* is an extension of CLF supporting a multi-party, "multi-directional", multi-attribute negotiation in the search phase of the execution of coordi-

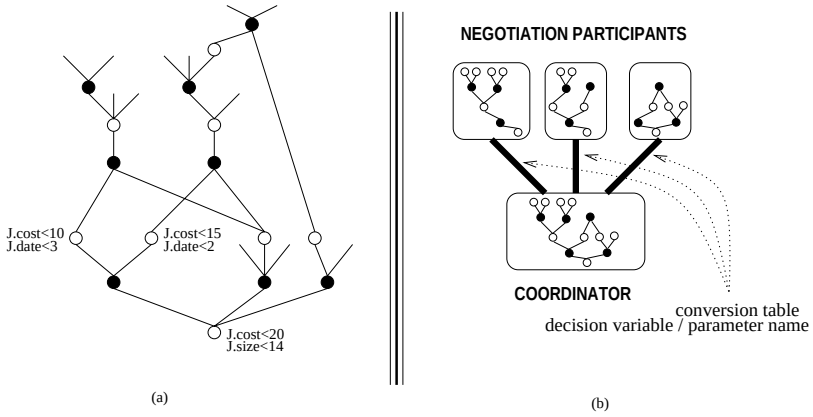

Figure 3. An example of negotiation graph and graphs copies across negotiation participants.

nation scripts. It allows the resources that trigger a rule to be negotiated by successive refinements between the components involved in the negotiation. The search tree thus becomes a "negotiation graph", which captures the dependencies between the negotiation interactions.

A negotiation graph is a directed bi-colored graph expressing the topological structure of a negotiation: white nodes characterize the contexts in which decisions are taken; black nodes characterize alternatives in a decision. Each context (white) node in the graph contains constraints on different issues for the parameters of the service execution that is being negotiated. For example, for the services `outsrc(job)` and `insrc(job)`, the (here unique) parameter of the negotiation (`job`) is a print job and an issue can be the price which can assume a range of possible values. Different branches of negotiation can be created in the negotiation graph to explore alternatives in terms, say, of price. The partners may then refine each branch specifying different delays. The interaction specifying the delay would occur in the context of one or the other branch created by the interaction concerning the price. Fig. 3(a) shows an example of a negotiation graph.

A negotiation process is modeled as the collaborative construction of a negotiation graph among the negotiation participants. Each participant has its own (partial) copy of the negotiation graph (see Fig. 3(b)) and expresses negotiation decisions manipulating that copy. For example, a proposal made by the printshop who initiated the negotiation is represented in the graph copy visible by the `outsrc` service and the proposals made by printshops involved in the negotiation are represented in the graph copies visible through their `insrc` services. The purpose of the *CooF* protocol is to allow the synchronisation of the different copies (Andreoli and Castellani, 2001). So, the partners do not communicate directly, but only via a set of operations, that

they perform on their negotiation graph copies and which are appropriately propagated by the *CooF*. The **Connect** operation allows to involve a new participant in a negotiation. The **Assert** and **Open** operations allow a participant to build the negotiation graph, by creating and populating nodes with information about the negotiation state at these nodes. The **Request** operation allows participants to express their information needs on given aspects in order to proceed in the negotiation. The **Ready** and **Quit** operations allow a participant to declare respectively that it is "ready to sign" in the state of a given negotiation context, or that it wishes to leave a given negotiation branch at that state.

9. Example

We give two examples of negotiation in E-Alliance detailing the interactions between managers, *NegFs* and the *CooF*. We consider an e-alliance including a coordinator (C) and three printshops (A0, A1, and A2). Let's assume that the manager of A0 wants to *outsource a job as a block* to one of two candidates, A1 and A2. He specifies then a negotiation framework and an object:

```
(Context :duration 1min :messages 100 :candidates 10 :contractants 1
         :tactics as_a_block :protocols (:manager inform :negf ICNIP))

(ON0 :type BWP :contractants (A1 A2)
     :job ((size ? (size==50) pref_size) (cost ? (cost<=200) pref_cost)
          (quality ? (quality enum {high,low}) pref_quality))
          (date ? (date <= 3) pref_date)
          (penalty ? (penalty <= 40) pref_penalty)) (...) >>
     :dependencies ((quality low ==) (cost 140 <=) if) (penalty (cost 10 /) =))
```

The "as a block" tactic, described in Sec. 7, can be modeled by the following rule, interpreted by C, whose execution corresponds to the activation of the `outsrc` service in A0 and of the `insrc` service in A1 and A2:

```
outsrc(job) @ asablock(job) @ partner(d,job) @ insrc(d,job)
<>- transfer(d,job)
```

The tactic for the job to be outsourced by splitting it into slots is represented by the rule:

```
outsrc(job) @ split(job,job1,job2) @ partner(d1,job1) @
partner(d2,job2) @ insrc(d1,job1) @ insrc(d2,job2)
<>- transfer(d1,job1) @ transfer(d2,job2)
```

A0's *NegF* starts negotiating with A1 and A2 using the ICNIP protocol:[1]:

[1] The ACL used here allows to specify the messages routing (:com), the definition of the interaction (:mas), and the content of the negotiation object (Carron et al., 1999).

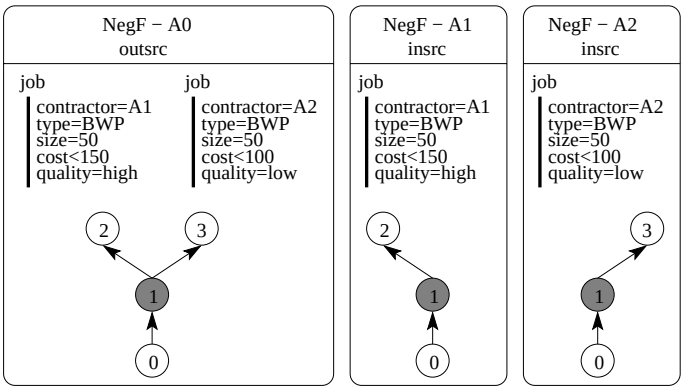

Figure 4. Negotiation with a *as a block* tactic.

```
(message :com (:sender A0 :receiver A1 ...) :mas (:act cfp ....)
         :content (obj :type BWP :job (size ? (size == 50K))
         (cost ? (cost<=150)) (quality ? (qualify enum {high}))))
(message :com (:sender A0 :receiver A2 ...) :mas (:act cfp ....)
         :content (obj :type BWP :job (size ? (size == 50K))
         (cost ? (cost<=100)) (quality ? (quality enum {low}))))
```

These interactions correspond, at the middleware level, to the creation of two white nodes in the negotiation graph managed by the service `outsrc` of A0 and propagated as the creation of a white node in each negotiation graph managed by the `insrc` services of A1 and A2 (Fig. 4). A1 and A2 have only a partial view of the complete negotiation graph (the one which is relative to the propositions sent to each of them by A0) whereas A0 has a global view. When A1's and A2's *NegFs* receive these messages, they consult their respective managers and build the negotiation object and framework that they will use to manage the negotiation with A0. From them, they generate different proposals which will be communicated through the *CooF* by opening nodes in the negotiation graphs of A0's `outsrc` service and in the negotiation graphs managed by the respective `insrc` service of A1 and A2. The negotiation goes on until A0 decides to commit on the proposition made by A1. Then a contract is formulated with the instantiated ON0 object by the *ConF*.

Let's suppose now, that the tactic of the initial negotiation framework is to outsource the job *splitting* it. Starting from the negotiation object specified by its manager, the A0' *NegF* creates three objects: ON0 (analogous to the one for the previous case) and ON1, resp. ON2, for the negotiation with A1, resp. A2:

```
(ON1 :type BWP :contractants (A1) :job ((size ? (size <= 50) pref_size)...))
(ON2 :type BWP :contractants (A2) :job ((size ? (size <= 50) pref_size)...))
```

Upon reception of the two corresponding call for proposals, the *NegFs* of A1 and A2 consult their managers and build their negotiation objects and

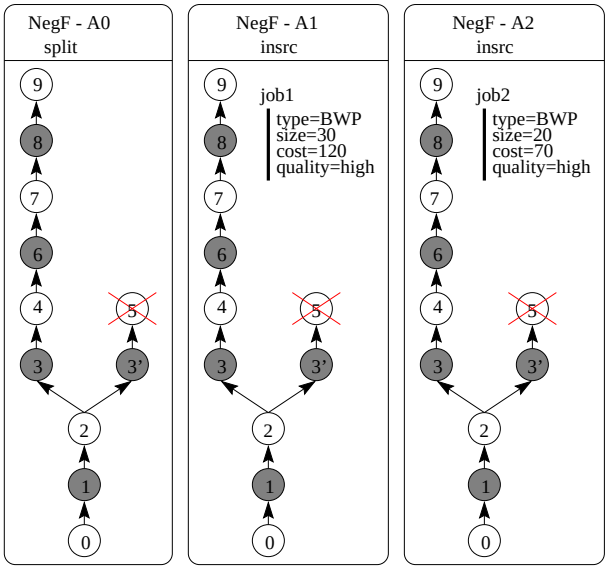

Figure 5. Negotiation with a *split* tactic.

frameworks. Proposals are communicated through the *CooF* by opening nodes in the negotiation graphs managed by the A0 `outsrc` and `split` services and in the negotiation graphs of the `insrc` services of A1 and A2. The A0 `split` service manages the link between ON0, ON1 and ON2: for each issue in ON1 and ON2, the constraints in the corresponding issue in ON0 must be satisfied. Therefore, the negotiation graphs managed by the `insrc` services of A1 and A2 have the same form but with different values on the node: each participant only knows the values of its own job (see Fig. 5). The A0 `split` service has a complete view of the negotiation expressed by ON0. For example, let's suppose that the negotiation reaches the status described in Fig. 5. Node "9" in the negotiation graph managed by `split` contains information associated to both ON1 and ON2. Both objects correspond to (ON0 :type BWP :contractants (A1,A2) :job (size 50) (cost 190) (quality high)). If the manager decides to accept the proposal, then two (inter-dependent) contracts will be "signed" in the adoption phase, one for each of the two instantiated ON1 and ON2 objects.

10. Related Work

The kind of alliances we consider are typical of virtual enterprises, e-business, and e-commerce networks. Several institutions, e.g. IST (Hurwitz, 1998), work on these topics and some proposals are available on the market, e.g. ROSETTANET. On the research side, several approaches have been proposed, e.g. in the MEMO EU project, for supporting inter-organisational co-

ordination. However, these environments focus on orders management among partners providing poor support to negotiate contracts or manage alliances.

In dynamic systems, like e-alliances, *uncertainties* (Wellman and Walsh, 1999) introduce several complications. The decomposition of the E-Alliance infrastructure in layers and facilities helps to manage them in a concurrent fashion: *AllF* - failures of agents, intentional fraud; *NegF* - variation in knowledge across agents, multiplicity of equilibria; *ConF* - withdrawal of contracts, renegotiation of failed contracts; *CooF* - asynchronous communications.

In the multi-agent community, a lot of work has been devoted to support negotiation (Dignum and Cortes, 2001). Many of these proposals support brokering or auction negotiation by matching buyer attributes priorities with the sellers available products. Our approach proposes a decentralised multi-issue negotiation model in which a set of agents can conduct *several* one-to-one conversations in a concurrent manner according to the same middleware protocol.

As in (Barbuceanu and Lo, 2001; Bui and Kowalczyk, 2001), we consider negotiation as a constraint based reasoning problem, proposing a formalisation of constraints over both the attributes values and different concurrent negotiations. Differing from systems that use dependencies between tasks using a hierarchical decomposition and treatment (Finin et al., 2000), in E-Alliance all negotiations are treated at the same level of importance and in the same concurrent and asynchronous manner.

Middleware systems are gaining momentum, trying to satisfy recurrent needs of distributed applications, esp. in the domain of e-commerce (Charles, 1999). The middleware layer for E-Alliance is designed to provide negotiation facilities accounted for at the lowest level, in the interaction protocol between the components of the system, and not as a side service, like in Corba (McConnell, 1999).

References

I. Alloui and F. Oquendo. Software Process Architecture = Process Components + Intention-based Interactions. In *Proc. of PDPTA*, Las Vegas, Nevada, 2001.

J.M. Andreoli, D. Arregui, F. Pacull, M. Riviere, J.Y. Vion-Dury, and J. Willamowski. CLF/Mekano: a Framework for Building Virtual-Enterprise Applications. In *Proc. of EDOC*, Manheim, Germany, 1999.

J-M. Andreoli and S. Castellani. Towards a Flexible Middleware Negotiation Facility for Distributed Components. In *Proc. of "E-Negotiations" (DEXA)*, Munich, Germany, 2001.

J.M. Andreoli, S. Castellani, and M. Munier. AllianceNet: Information Sharing, Negotiation and Decision-Making for Distributed Organizations. In *Proc. of EcWeb*, Greenwhich, U.K., 2000.

M. Barbuceanu and Wai-Kau Lo. Multi-attribute Utility Theoretic Negotiation for Electronic Commerce. In *AMEC III*, LNAI 2003, pp. 15-30.

V. Bui and R. Kowalczyk. On constraint-based reasoning in e-negotiation agents. In *AMEC III*, LNAI 2003, pp. 31-46.

T. Carron, H. Proton, and O. Boissier. A Temporal Agent Communication Language for Dynamic Multi-Agents Systems. In *Proc. of 9th MAAMAW*, LNAI 1647, 1999.

J. Charles. Middleware Moves to the Forefront. *IEEE Computer Magazine*, 32(5):17–19, 1999.

F. Dignum and U. Cortes, eds. *AMEC III*, LNAI 2003. Springer Verlag, 2001.

P. Faratin. *Automated Service Negotiation Between Autonomous Computational Agents*. PhD thesis, Dep. of Elec. Engin. Queen Mary & Westfield College, 2000.

T.Finin Y.Chen, Y.Peng and Y.Labrou. A negotiation-based multi-agent system for supply chain management. *Issues in Agent Communication*, pp. 178-192, 2000.

P. Grefen and S. Angelov. B2B eContract Handling – A Survey of Projects, Papers and Standards. TR-01-21, CTIT, University of Twente, 2001.

J. Hubner, J.S. Sichman, and O. Boissier. A model for the structural, functional, and deontic specification of organizations in multiagent systems. In *Proc. of SBIA'02*, LNAI 2507, pp. 118128, Porto de Galinhas, PE, Brazil, 2002.

S.M. Hurwitz. W.P.: Interoperable Infrastructures for Distributed Electronic Commerce. Nat. Inst. of Standards and Technology, http://www.atp.nist.gov/atp/98wp-ecc.htm, 1998.

R. Keeny and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. John Willey & Sons, 1976.

S. McConnell. Negotiation Facility. Technical Report, OMG, 1999.

ProcessWise Integrator PML Reference Manual. ICL/PW/635/0, Versions 4.1-4.5, April 1996.

J.S. Sichman, R. Conte, Y. Demazeau, and C. Castelfranchi. A social reasoning mechanism based on dependences networks. In *Proc. of ECAI*, Amsterdam, The Netherlands, 1994.

L. Vercouter. A distributed approach to design open multi-agent systems. In *Proc. of ESAW*, 2000.

M.P. Wellman W.E. Walsh. Modeling supply chain formation in multiagent systems. *in IJCAI-99 Workshop on Agent Mediated Electronic Commerce*, 1999.