MULTI-AGENT-BASED SUPPORT FOR ELECTRONIC CONTRACTING IN VIRTUAL ENTERPRISES

Benjamin Gâteau [*,**] **Olivier Boissier** [**]
Djamel Khadraoui [*]

[*] *CRP Henri Tudor, Luxembourg*
[**] *ENSM Saint-Etienne, France*

Abstract: In this paper we present multi-agent based management of electronic contract (e-contract) in a B2B context, for Tele-services. Several solutions have been proposed in the multi-agent domain for negotiation of contracts. However few concern the monitoring of contracts and the management of their violation and enforcement. We describe a prototype implementing the concepts of e-contract using $\mathcal{M}$OISE^{Inst}, a normative multi-agent organizational model. Thanks to it, contractual clauses are expressed as rights and duties of agents along four points of view: structural, functional, contextual and normative. We use a real-world contracting scenario as a test-bed for examining the proposed e-contract architecture and for implementing our prototype. *Copyright © 2006 IFAC*

Keywords: Distributed artificial intelligence, Agents, Norms, Model, Constraints.

1. INTRODUCTION

With the development of the Internet and Virtual Enterprises appearance, e-Services and e-Contracting in general have gained an increasing interest in the business domain. While some B2B applications handle electronic contracts (e-contract) and digital signatures as digitalized paper contracts (Laurikkala and Tanskanen, 2002), an increasing number of them aim at increasing their enactment and management (e.g. (Koetsier *et al.*, 2000)).

In this global trend, multi-agent technologies have been introduced to achieve the automation in creation, execution and monitoring of e-contracts by agents on behalf of users. Lots of work deal with negotiation of the issues related to the content of contract and to their execution(e.g. (Dignum and Sierra, 2001)). The resulting contracts consist in digital agreement between contractual parts where rights and duties in terms of deliverables, costs and delays of the participants are explicitly represented. However such contracts often lead to inflexible relations between participants. The obtained result is in contrary to the requirements of open and dynamic system that are stressed by the actual business paradigms aiming at improving the competitiveness of companies like dynamic virtual enterprises and dynamic service outsourcing (Hoffner *et al.*, 2001). Moreover, few of the existing research works take into account the monitoring of contracts clauses (Padovan *et al.*, 2002) that bind agents together.

The aim of our work is to define an e-contract infrastructure that provides the same guarantees and proofs that we can find in classic contract management, with more security and flexibility (negotiation before and renegotiation during the execution). In this paper we are considering

EBSME [1] (Khadraoui and Dubois, 2003), a web-based tele-service application. This application mediates the interactions between two kinds of users – employers and employees – for trading translation services, intangible assets, involving both parties: employers seeks for employees whose skills and knowledge match the requirements for realizing the proposed job. Agreement between parties are represented as e-contracts, description of the duties (obligation, interdictions) and rights (permissions) between an *employer* and *employees* for realizing the job in terms of deliverables, costs and deadlines. EBSME coordinates the registration, negotiation/creation, signature and execution of secured e-contracts concerning those tele-services between both parties (cf. Fig. 1). In order to automate the execution and monitoring of e-contracts, multi-agent technologies have been used. Users are thus able to delegate to the application a part of their control in the management and monitoring of contracts. To this aim, we have enriched and formalized the e-contract model to enable an agent-based execution and management of them on one hand, and, on the other hand, we have added a supervision system to control and enforce the contracts between employer and employees.

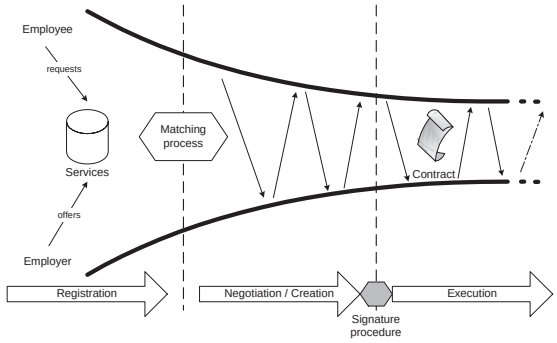

Fig. 1. EBSME: global view of contract management steps

This paper is organized as follows. We describe first $\mathcal{M}$OISE^{Inst}, a Multi-Agent organizational model that we use to model contracts in our application. We then present the global architecture of the system and the main step of our approach. The following section is dedicated to the presentation of the implemented architecture with the contract model applied to the scenario of the employment domain.

2. FOUNDATIONS

In this work we propose a multi-agent support for the enactment and monitoring of the different tasks specified in the contract for realizing it (see *contract execution* in Fig. 1). Moreover, we have added support for enforcement that could take place during the execution of contracts in case of its violation. Contracts specify the agreement between employees and employer concerning their participation to the distributed execution of the job. A contract must describe both the functioning and the structure organizing this functioning. Moreover it contains explicit legal dimensions bearing on the involved participants. In order to take this into account we propose to use a "normative organizational model", called $\mathcal{M}$OISE^{Inst}, to express e-contract. This normative organizational model is accompanied by a specialized "normative middleware", called $\mathcal{S}$YNAI [2] to monitor and enforce legal aspects expressed in the contracts.

2.1 Normative Organization Modelling

$\mathcal{M}$OISE^{Inst} (Gateau *et al.*, 2005) is founded on the $\mathcal{M}$OISE^{+} organizational model [3] (Hubner *et al.*, 2002). It is composed of the following components that are used to specify an organisation of agents in terms of structure, functioning, evolution and norms (see Fig. 2):

- A *Structural Specification* (SS) defines: (i) the *roles* that agents will play in the organization, (ii) the *relations* between these roles in terms of authority, communication or accointance, (iii) the *groups*, additional structural primitives used to define and organize sets of roles ;
- A *Functional Specification* (FS) defines global *business processes* that can be executed by the different agents participating to the organization according to their roles and groups;
- A *Contextual Specification* (CS) specifies, a priori, the possible evolution of the organization in terms of a *state/transition graph* ;
- A *Normative Specification* (NS) defines the deontic relations gluing the three independant specification (SS, FS, CS). This NS clearly states rights and duties of each roles/groups of SS on sets of goals (missions) of FS, within specific states of CS.

A BNF [4] definition of OS is available in (Gateau *et al.*, 2004) for SS and FS and in (Gateau *et al.*, 2005) for CS and NS.

2.2 Normative Organization Middleware

In human societies, institutions define rules (North, 1990) that enclose all kinds of formal or infor-

[1] EBSME: Electronic Business for the Small and Medium Enterprises

[2] $\mathcal{S}$YNAI: SYstem of Normative Agents for Institution

[3] $\mathcal{M}$OISE^{+}: Model of Organization for multI-agent System

[4] BNF: Backus-Naur Normal Form

mal constraints that human beings use to interact. These last years, electronic institutions have been introduced in multiagent domain and more particularly in agent-mediated e-commerce. They propose to model these rules with normative systems (Jones and Carmo, 2001). Institutions are defined as a set of agents, which behave according to norms taking into account their possible violation. The functioning of the agents is supervised and controled with a set of *institution services* (e.g. (Dellarocas, 2000)). The institution services that support and use $\mathcal{M}$OISE^{Inst} are regrouped in a specific "normative middleware" called $\mathcal{S}$YNAI on which the agents execute. This layer is in charge of: (i) managing the life cycle of SS as entering/exiting of agents within the organization, or requesting/leaving of roles or groups by the agents, (ii) coordination of the concurrent execution of FS as commitment to missions or achievement of goals, etc, (iii) dynamic and evolution of the organization state through the CS, (iv) the monitoring and supervision of norms of NS activated/deactivated by the evolution of the organization.

Using both $\mathcal{S}$YNAI and $\mathcal{M}$OISE^{Inst}, we are able to define an e-contract management system as a set of agents whose behavior is ruled by contracts expressed as organizations expressed in $\mathcal{M}$OISE^{Inst} and controlled by an arbitration system, implemented with $\mathcal{S}$YNAI, that has the possibility to reward or to punish agents in case they respect or not their agreements as expressed in e-contracts. Using $\mathcal{M}$OISE^{Inst}, EBSME agents are able to reason and to take into account the different e-contracts described with $\mathcal{M}$OISE^{Inst}. The $\mathcal{S}$YNAI platform on its side take into account this specification in order to supervise and control EBSME agents. Both layers are based on an agent execution platform.

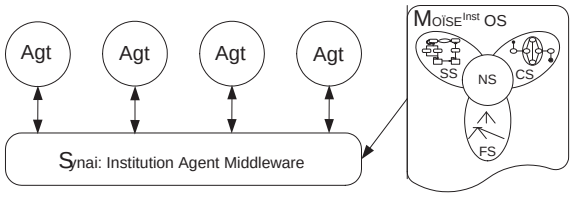

Fig. 2. $\mathcal{M}$OISE^{Inst}, normative organization specification language, and $\mathcal{S}$YNAI, Normative middleware.

These four specifications form the Organizational Specification (OS). The Organizational Entity (OE) is then built from the set of agents that have adopted a role according to the SS of the OS. From this time the $\mathcal{S}$YNAI middleware manages and controls the functioning of this OE (composed of the entities -states- of every specification NE, CE, SE and FE) by the way of different events corresponding to the entry/exit of agents of the OE, adoption/leaving roles or groups, change of context, commitment to missions, achievement of goals.

3. EBSME E-CONTRACT MANAGEMENT ARCHITECTURE

EBSME is a multiagent system organised along four layers: (i) *e-Contract Management*, man machine interfaces where users can interact with the system to manage their contracts, (ii) *e-Contract Execution* that enacts and executes e-contracts on behalf of their parties, (iii) *e-Contract Supervision*, that supervises and enforces if necessary the e-contract execution done in the above layer (iv) *agent-execution platform* that manage the distribution and exchange of messages between agents. In the following we will not describe the two extrem layers since not being in the main focus of this paper. The agent-execution platform is implemented using SACI (see (Hubner and Sichman, 2000) for further details) and the e-Contract management layer is detailed in (Khadraoui and Dubois, 2003) for the initial architecture and in (Khadraoui and Gateau, 2005) for the multiagent based architecture. Both middle layers are implemented with two kinds of agents:

- on the e-Contract Execution Layer, personnal e-contract management agents (PeMA) consult, activate and deactivate their e-contract portfolio on behalf of their user.
- on the e-Contract Supervision Layer, institution e-contract management agents (IeMA) as stated in $\mathcal{S}$YNAI middleware that supervise and enforce the execution of the PeMA as specified in contracts.

PeMA play the roles and goals as expressed in their active e-contracts (see sec. 4). They interact with each other to coordinate and manage the execution of current e-contracts. Each PeMA aims at controlling the actions carried out by the users, reading/writing in the contract database (portfolio), taking the user informed of the evolution and state of the contracts to which they participate. They may also have the possibility of making decisions regarding the fulfillments or not of the contracts.

Dedicated IeMA are attached to each active e-contract: a "NEManager" that supervises the normative state of the e-contract as specified in the e-contract NS, an "SEManager" that manages the distribution of all the roles among the parties involved in the e-contract as specified in SS, a "FEManager" that manages the current state of the execution and coordination of the FS of the e-contract, a "CEManager" that manages the current context of the e-contract as specified in CS,

an "InstArbitrator" that grants a reward and a penalty according to the sanctions specified in e-contract NS and violations detected by "NsManager". After negotiation and e-contract creation, an "InstManager" checks and stores all the current active e-contracts in the system. Finaly an "OrgFacilitator" is attached to each PeMA in order to help it to interact with the agents of $\mathcal{S}$YNAI. Besides these IeMA used for the enforcement and execution of e-contracts, we also have defined a "Matchmaker" that matches the requested services to the provided services already present in the system (when entering the system, an agent publishes its provided and requested services in order to be registered). A "Reputation" agents allows PeMA to know the reputation of other agents in order to decide if they can trust them. In the current architecture, this IeMA is represented by a value, allocated to each agent by the "InstArbitrator", based on the agents' execution contract results.

4. EBSME E-CONTRACT SPECIFICATION

E-contract specification in EBSME is based on the $\mathcal{M}$OISE^{Inst} meta-model presented in the previous section. It contains the explicit definition of the participants, products and services on one hand, and, on the other hand a contract template stating the business process that should be enacted to fulfil the contract, the different obligations, interdictions and permissions that users have to respect to comply to the contract. All these aspects are expressed using the four specification of $\mathcal{M}$OISE^{Inst}. In our context a contract consists in a unique identifier (:id), a header (:header), a set of general conditions (:conditions), a structural specification (:SS), a functional specification (:FS), and a normative specification (:NS). Others contract templates could be defined with other options (more participants, different execution contexts, etc.)

⟨Contract⟩ ::= '(' Contract :id ⟨cId⟩ :header ⟨header⟩ :conditions ⟨condition⟩* :SS ⟨SS⟩ :FS ⟨FS⟩ :NS ⟨NS⟩ ')'

The setup of a contract template into an e-contract is realized through the definition of the header and of the general conditions which will define the context of execution. Both sets instantiate the structural, functional and normative specifications that describe the elements of the contract template, the organization of the participants, the clauses and finally the methods of execution of the contract.

4.1 Header

The header specifies the participants, the services and the products involved in the contract, ⟨header⟩ ::= '(' :participant ⟨participant⟩* :product ⟨product⟩* :service ⟨service⟩* ')', consisting in : (i) a list of couples of agent's identifiers and roles from the SS, mapping agents with the roles they play into the contrat (:participant), (ii) a list of service's identifiers and attributes concerned by the contract expressed as goals appearing in the FS as part of the business process (:service), (iii) products concerned by the contract referencing the FS and described with a set of attributes (:product). Values of product and service attributes may have been defined during negotiation.

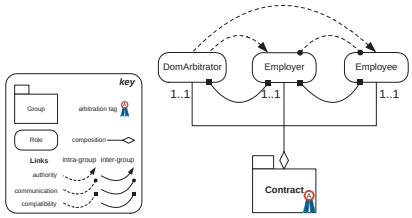

Fig. 3. e-contract: structural specification

4.2 Structural and Functional Specifications

The SS (cf. Fig. 3) defines the structure of the e-contract template (group "Contract"). It is composed of three roles ("Employer", "Employee", "DomArbitrator") played at most by three agents (cardinality on composition link) in the context of a concrete realization of the contract template (roles belongs to the "Contract" group). The links between the roles give them the right to communicate with each other. The role "DomArbitrator" has authority over the two other roles, enabling control of agent playing the first role on the others.

The FS of the e-contract template specifies the global tasks that are concerned by the e-contract. They are expressed as a set of missions ("mUser", "mArb", "mEe", "mEr") dealing with the realization of the services and of the products defined in the :header. In our work (see Fig. 4) , the FS consists in the following social schemes : "Contract execution" and "Deliverable execution" schemes and a "Sanction scheme".

- "Contract execution" and "Deliverable execution" schemes define a business process expressed as one or more goals to produce the services and products involved in the contract.
- The "sanction" scheme defines a dedicated business process related to the sanctions enforcing in the system. Sanctions are expressed in the NS part of the contract, as decrease of reputation or payment.

This splitting in those two kinds of social schemes allows a separate specification of normal scenarios of the contract and exception scenarios (in the case of non-respect of deontic relations).

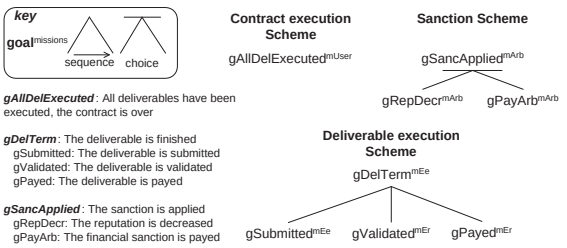

Fig. 4. e-contract: functional specification

4.3 Contextual and Normative specifications

The contextual specification describes the evolution of the contracts in the organization as well as the rules of transition from one context to the other. In this application, the dynamic of contract being not studied, we reduce the CS to its simplest form, i.e. a single state which stays active all during the life cycle of the e-contract.

The normative specification of the contract expresses rights and duties of the participants in terms of obligations, permissions and prohibitions. We have associated these deontic expressions with sanctions (penalties and rewards) in order to control agents behaviors for the execution of the contract. This part of the contract establishes the link between the roles defined in the SS and the missions defined in the FS according to the CS. Thanks to it, we are able to express that an agent playing a specific role ought to or is not allowed to execute a specific mission. The representation of sanctions enables each agent participating to the contract, to reason on the consequences of the non-observance of an obligation.

In $\mathcal{M}$OISE^{Inst}, a norm will define a right or a duty for a role or a group to execute a mission in a particular context and during a given time supervised by an issuer which can apply a sanction if the norm is not respected. We formally define a norm as follows: ⟨norm⟩ ::= '(Norm ':id ⟨normId⟩ :weight ⟨int⟩ '::' [:conditions ⟨condition⟩] :operator ⟨deonticRel⟩ :bearer ⟨sentityId⟩ :issuer ⟨sentityId⟩ :context ⟨contextId⟩ :action ⟨deonticAct⟩ [:relation ⟨relation⟩ :deadline ⟨date⟩][:sanction ⟨normId⟩]')'

id	*condition*	*bearer*	*deOp*	*mission*	*deadline*	*sanction*
N01	*null*	Employer	O	mUser	< *end*	N05
N02	*null*	Employee	O	mUser	< *end*	N05
N03	*null*	Employee	O	mEe	< *end*	N05
N04	*null*	Employer	O	mEr	< *end*	N05
N05	violated(?N)	DomArbitrator	O	mArb	*null*	*null*

Fig. 5. e-contract: normative specification

Given this NS definition, we are thus able to specifiy that roles "Employer" and "Employee" have the obligation to execute mission "mUser" before the end of contract with the definition of two norms "N01" and "N02" (cf. Figure 5). Norm N03 expresses at its turn a dedicated clause for role "Employee" stating that it ought execute mission "mEe" before a certain deadline. The sanction N05 that appears within each of the previous norms is expressed a norm beared by the "DomArbitrator" role as an obligation to execute mission called "mArb" without any time constraint. On the Figure 5, Context is not mentioned because it is specified at *null* for each norm, the same for Weight which is specified at *1* and for Issuer specified at *Supervisor*.

As we can see, in this specification, different fields express the binding of the different specifications with each other:

- The :bearer field refers either to a role or a group of the SS on which the norm is applied. When the :bearer is a group, all roles belonging to this group in the SS, are the :bearer of this norm. A norm is also defined toward either a role or a group expressed in the field :issuer. The :issuer defines the entity which has the responsibility to control if the norm is not violated. If the :issuer is a group then all roles that belong to the group have the responsibility to detect a violation and to trigger a sanction.
- the field :context refers to the particular context of the CS in which the norm must be considered.
- the field :action connects missions of the FS to the :bearer of the norm. A :deadline may specify when the norm is valid: before ('<'), when/while ('=') or after ('>') a date expressed by the :relation field.

  ⟨deonticAct⟩ ::= 'do''('⟨missionId⟩')' | ⟨actionId⟩
  ⟨relation⟩ ::= '>' | '<' | '='

- Finally, the field :sanction refers to a norm in the NS itself. It expresses a "sanction" to apply in case of norm violation.

In addition a norm is specified with:

- A :condition expressing the particular state of the OE in which the norm may be applied. Tests on the state of the OE are expressed by functions ⟨function⟩ referring to global variables that are accessible by the :issuer and :bearer of the norm. For instance, we have defined the predicates to test if a norm is violated (*violated*) or not (*respected*). We can know how much agents are part of a group (*number*) and what is the maximum of agents that a group accepts (*cardinalityMax*). ⟨condition⟩ ::= '('(⟨condition⟩ 'AND' ⟨condition⟩) | (⟨condition⟩ 'OR' ⟨condition⟩) | (⟨function⟩ ['!'] '=' ⟨value⟩)')' ⟨function⟩ ::= 'violated('⟨normId⟩')' | 'respected('⟨normId⟩')' | 'number('⟨groupId⟩')' | 'cardinalityMax('⟨groupId⟩')'
- The field :operator defines if the norm is an obligation (O), a permission (P) or a prohibition (F).
- Finally, the :weight field defines a priority level used for solving conflicts between norms,

when for instance an agent could be constrained by two contradictory norms. In case of two norms, with the same :weight then the agent has to solve the conflict by taking into account its internal priorities that it can base on the :sanction attached to the norms.

As in SS and FS, in a NS, norms may be separated into those that are relevant to the institution and those that are relevant to domain. A norm which is relevant to the institution is a norm issued from a role belonging to the institution part of the SS. Such a norm is used for controlling the respect and sanctions applied to the "domain" agents.

After the signature, the contract is verified by the "InstManager" and stored into the database where it could be used by the other IeMA. From the EBSME point of view, the contract is executed by modifying the state of the associated deliverables. Relating to the MAS framework and contract model, each deontic relation is associated with a deliverable. When a deliverable is provided a commitment is fulfilled. The deliverable could be accepted or rejected by the consumer. If the deliverable is rejected, a conflict appears and then the "DomArbitrator" decides if the rejection is justified or not. This case is described in the sanction social scheme defined in the contract ⟨FS⟩ part.

5. CONCLUSION

A prototype of the EBSME application has been implemented and tested using the SACI platform on which PeMA and IeMA agents have been deployed and bounded to J2EE infrastructure for execution of services and products as EJB. (Khadraoui and Gateau, 2005) presents this prototype with a description of the multi-agent organized with the specification detailed here. This application is used as platform for validating our e-contract model and study how it can help in discharging the physical user from the constraints of monitoring his contracts.

REFERENCES

Dellarocas, Chrysanthos (2000). Contractual agent societies: Negotiated shared context and social control in open multiagent systems. In: *Workshop on Norms and Institutions in Multiagent Systems*. Barcelona.

Dignum, Frank and Sierra, Carles, Eds. (2001). *Agent-Mediated Electronic Commerce - The European AgentLink Perspective*. Vol. 1991 of *LNAI*. Springer Verlag.

Gateau, Benjamin, Djamel Khadraoui and Eric Dubois (2004). Architecture e-business sécurisée pour la gestion des contrats. In: *3ème Conférence sur la Sécurité et Architectures Réseaux (SAR)*. La Londe - France.

Gateau, Benjamin, Olivier Boissier, Djamel Khadraoui and Eric Dubois (2005). $\mathcal{M}$OISE^{Inst}: An organizational model for specifying rights and duties of autonomous agents. In: *International Workshop on Coordination and Organisation*. Namur - Belgium.

Hoffner, Yigal, Simon Field, Paul Grefen and Heiko Ludwig (2001). Contract-driven creation and operation of virtual enterprises. *Comput. Networks* **37**(2), 111–136.

Hubner, Jomi Fred and Jaime Simao Sichman (2000). Saci: Uma ferramenta para implementaão e monitoração da comunicação entre agentes. In: *7th Ibero-American Conference on AI*. São Carlos. pp. 47–56.

Hubner, Jomi Fred, Jaime S. Sichman and Olivier Boissier (2002). $\mathcal{M}$OISE^{+}: towards a structural, functional, and deontic model for mas organization. In: *International Joint Conference on Autonomous Agents and MultiAgent Systems*. Bologna, Italy. pp. 501–502.

Jones, A. and J. Carmo (2001). *Handbook of Philosophical Logic*. Chap. Deontic logic and contrary-to-duties, pp. 203–279. Kluwer.

Khadraoui, Djamel and Benjamin Gateau (2005). A prototype for an agent-based electronic contracting using an organizational model. In: *International Conference on Intelligent Agents, Web Technologies and Internet Commerce*. Vienna, Austria.

Khadraoui, Djamel and Eric Dubois (2003). B2B eContract solution for teleservices. In: *International Conference on Intelligent Agents, Web Technologies and Internet Commerce*. Vienna, Austria.

Koetsier, Marjanca, Paul W. P. J. Grefen and Jochem Vonk (2000). Contracts for cross-organizational workflow management. In: *International Conference on Electronic Commerce and Web Technologies*. London, UK. pp. 110–121.

Laurikkala, Heli and Kari Tanskanen (2002). Managing contracts in virtual project supply chains. In: *IFIP TC5/WG5.5 Third Working Conference on Infrastructures for Virtual Enterprises*. The Netherlands. pp. 93–100.

North, Douglass C. (1990). *Institutions, Institutional Change and Economic Performance*. Political Economy of Institutions and Decisions. Cambridge University Press.

Padovan, Boris, Stefan Sackmann, Torsten Eymann and Ingo Pippow (2002). A prototype for an agent-based secure electronic marketplace including reputation tracking mechanisms. *International Journal of Electronic Commerce* **6**(4), 93–113.