Programming institutional semantic agents

JIA: an institutional extension for JADE semantic agents

Tracking number 118

ABSTRACT

JADE is a development platform to design agents able to exchange speech acts in the FIPA standard. JSA is a rule-based interpretation engine grounding on a BDI logic allowing JADE agents to semantically interpret the FIPA speech acts that they exchange. In this paper we describe an extension for JSA allowing semantic agents to reason about institutional concepts (institutional truth, count as rules, institutional powers) and to exchange and interpret institutional speech acts (declarations, promises).

Categories and Subject Descriptors

I.2.11 [**Artificial intelligence**]: Distributed Artificial Intelligence—*Intelligent Agents*; J.1 [**Computer applications**]: Administrative data processing—*Business*; K.4.4 [**Computers and Society**]: Electronic Commerce—*Distributed commercial transactions*

General Terms

Experimentation

Keywords

Social/Organizational Structure, Java Agent DEvelopment Framework, JADE Semantic Add-on, business to business

1. INTRODUCTION

In new applications of multi-agent systems, agents must be able to respect some rules of behaviour. For now they have been forced to do so by strict constraints impeding them to perform forbidden actions. For example agent communication languages like FIPA [4] make a strong sincerity hypothesis, and agents using this language are technically unable to lie. JADE (Java Agent DEvelopment Framework, [5]) is a framework to develop multi-agent systems in compliance with the FIPA specifications. JSA (JADE Semantics Add-on, [7]) is a framework to develop cognitive agents in compliance with the FIPA-ACL formal specifications. Now we want agents to be aware of rules, but to be able to decide autonomously if they want to respect them or not, as would human beings do. We have thus implemented an institutional extension for JSA, by grounding on Demolombe and Louis' logic of norms, roles and institutional powers [3].

This extension is called JIA (JSA Institutional Add-on). It is compliant with FIPA but it enriches this standard with new institutional concepts, allowing agents to reason about their obligations and the other agents' ones, and to detect and react to violations of institutional rules... Agents can learn new rules and adapt their behaviour subsequently. The underlying logical framework ensures the correctness of the agents' reasoning.

In this paper we begin with presenting an interesting industrial application field for agents able to behave correctly in an institutional context: the automation of business to business exchanges. We then describe the architecture and functioning of JIA.

Cite as: Programming institutional semantic agents, Tracking number 118, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

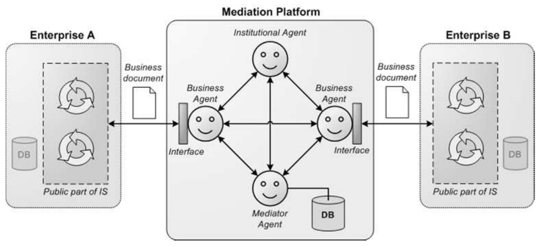

Figure 1: Architecture of the B2B mediation platform

2. APPLICATION

2.1 Context

An interesting application field for institutional agents is the managing of interoperability problems in automatic business to business (*B2B*) exchanges. Indeed enterprises all have their specific business process constraining their interaction with other enterprises. But this interaction procedure may not match the one of the other enterprise, and it would be too costly to redefine it each time the enterprise has to interact with a new partner. Now B2B exchanges are constrained by institutional laws: the country laws ruling business, and the rules of the contract between the two partners. Thus institutional agents, able to reason about such laws, may be useful in this context.

We have thus used JIA to develop an application, namely a mediation platform for automated exchanges between businesses [1].

2.2 Agents

The platform is made up of the following agents:

- An agent representing each enterprise
- Agents representing the involved banks
- An interface agent for each enterprise, allowing its representative agent to communicate with the mediation platform
- A mediator agent providing debugging strategies
- An institution agent monitoring the interaction and maintaining the registry of the institution

These agents evolve in a particular institutional context specified by the generic rules of business exchanges, as well as the specific rules negotiated in their contract of interchange.

2.3 Scenario

In our scenario, the interoperability problem concerns the delivery and payment of an order. The client enterprise's business process constrains it to wait delivery before paying, while the provider enterprise's one constrains it to receive payment before delivering. Now if the provider accepts an order from the client, both enterprises are obliged to act (client is obliged to pay, provider is obliged to deliver), but none can act. We call this an *interblocking* situation: two actions are obliged but each one is blocked until the other one is performed.

This is where our mediator agent can intervene. His library of debugging strategies contain a strategy for this situation. This strategy consists in involving a third party bank that can loan money to pay the provider, making him able to deliver; the delivery then allows client to pay, and he reimburses the bank.

The exchange is constrained by the institutional laws. As it goes on, new obligations can be generated, and agents adapt their behaviour subsequently. Finally the institution agent monitors the exchange and can detect and sanction violation of these rules.

3. PRINCIPLE

3.1 Principle of JSA

The JADE Semantic Add-on (JSA, [7, 6]) is a rule-based motor compliant with the BDI logic used to formalise the semantics of the speech acts of the standard FIPA-ACL [4]. It provides a set of basic interpretation rules accounting for the intentional dimension of these speech acts. Thus, agents developed with JSA (called *Semantic Agents*) are able to semantically interpret the exchanged speech acts. For example an agent j receiving an Inform about a proposition p from another agent i deduces that agent i believes that p and intends j to also believe p; other rules then lead j to believe p himself. But the reasoning of *Semantic Agents* can be parameterised by adding new interpretation rules or customising existing ones.

3.2 JSA features

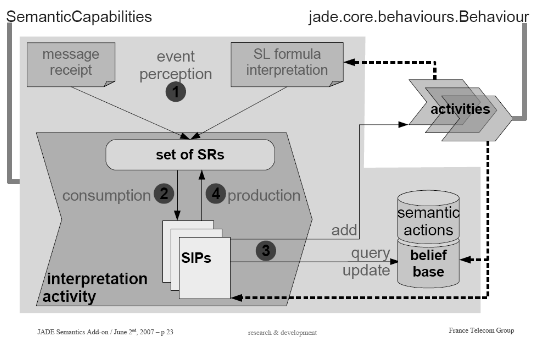

Figure 2: JSA interpretation engine

3.2.1 Knowledge Base

Semantic Agents have a knowledge base (below abbreviated as *KBase*) storing formulas representing their beliefs and intentions, and equipped with several mechanisms. **Observers** monitor formulas asserted in the *KBase* and react if their values change. **Filters** are of two types: **assertion filters** are applied before asserting a formula in the *KBase*, and may modify it, or may define a particular way to store it; **query filters** are applied when a formula is queried to the *KBase* and may modify or redirect the query. For instance an agent handling images may have an assertion filter storing images in external files (instead of storing a belief about their byte content, that would overflow its *KBase*), and a query filter to get the queried image from the external file where it is stored instead of from the *KBase*. Another simpler example is that of other agents' beliefs that are stored in nested *KBases* (one per other agent).

But before their storage in the agent's *KBase*, formulas go through a complex interpretation process.

3.2.2 Semantic Interpretation Principles (SIPs)

Each agent is equipped with a set of Semantic Interpretation Principles (*SIPs*). Their order of application is important, so the *SIPs* are numbered and stored in an ordered table.

Now the perceived events generate new Semantic Representations (*SR*), that are objects encapsulating a formula representing the event along with some attributes like possible annotations and an interpretation index. This interpretation index is used to determine which SIP will next interpret the *SR*. Each *SR* in the generated list is successively interpreted by the agent's *SIPs*, depending on its index.

Each SIP has a specific pattern of formula, and only deals with the incoming *SR* if its encapsulated formula matches this pattern. It can consume the incoming *SR*, modify its interpretation index in order to impact its subsequent interpretation, produce new *SRs*, or perform any kind of processing. As an output it generates a new list of *SRs*. If this list is empty the interpretation process is over and nothing is asserted in the *KBase*. Otherwise the *SRs* in the list are further interpreted by the next SIP indicated by their interpretation index.

The interpretation is over when all formulas have an interpretation index bigger than the number of the last SIP of the agent. The resulting list of *SRs*, possibly modified by this interpretation process, is finally asserted in the agent's

KBase (see figure 2).

For example Semantic Agents have a Belief Transfer SIP implementing the sincerity hypothesis (agents trust each other). This SIP deals with beliefs of other agents (*e.g.* interpreting agent i believes that agent j believes that φ) and converts them into beliefs of the interpreting agent (*e.g.* agent i adopts the belief that φ). This SIP can also be customized so that the agent trusts only some given agents and does not trust others.

3.2.3 Actions

Each agent is also equipped with a table of actions describing the actions he knows, with their preconditions and effects. He can only use known actions in his planning and reasoning. The default table contains at least all FIPA-ACL communicative acts.

3.2.4 Semantic capabilities

The Semantic Capabilities of an agent gather a set of the previous features: a table of Semantic Interpretation Principles, a table of Semantic Actions, and a *KBase* equipped with some filters and observers. These features define the agent's reasoning and behaviour.

3.3 Principle of institutional extension for JSA

Now we have developed an extension for the JSA, providing an additional set of interpretation rules to account for the institutional dimension of various types of actions: communicative actions of the FIPA standard, additional institutional communicative actions (declarations, promises), and application specific "material" actions. This extension called JIA allows to develop *Institutional Agents*, that are an extension of *Semantic Agents*, able to evolve in an institutional context. We have designed a generic set of rules that are needed to specify the basic behaviour of such agents. For example there are rules for interpreting new obligations, or violation of obligations by other agents... Now any designer interested in programming *Institutional Agents* can add new rules specific to his application, by specifying them in the declarative language described below (paragraph 3.3.1). The code allowing agents to deal with these rules is automatically generated.

3.3.1 JIA language

We have extended the FIPA-SL language used in JSA with the deontic and institutional operators defined in Demolombe and Louis' logical framework for norms, roles and institutional powers [3] (please refer to their paper for more details about this logic, since it is not the focus of this paper). We call this extended language xSL.

In particular we have operators to express the following notions:

- impersonal **obligations** are directed at states of the world that are obligatory, but no particular agent is responsible for them. Though the obligatory state can be that a given action has been performed by a given agent, in this case this agent is responsible for the fulfilment of this obligation;

- **institutional facts** are particular facts that cannot be physically observed, but that are considered true in the context of a given institution (for instance the fact that two people are married, the permission to drive a car when you have your license, or the duty to vote when you are of age;

- **normative consequences** (also known as *count as rules*) allow to deduce institutional facts from observable facts in the context of a given institution (for example in the law of a country, being of age counts as the permission to vote and the obligation to perform one's national military service);

- **institutional powers** are particular kinds of count as rules concerning the performance of actions. An agent has the power to establish a new institutional fact in a given institution if, in the context of this institution, his performance of a given procedure under a given condition *counts as* this new institutional fact. For example in the law of French Republic, mayors have the power to marry people by performing a declaration under some conditions like these people's agreement.

3.3.2 Institutional Capabilities

Institutional agents are endowed with Institutional Capabilities instead of Semantic ones. Actually, Institutional Capabilities are an extension of semantic ones, including the basic behaviour of semantic agents, but enriching it with new abilities in relation with the management of institutions: reaction to obligations, interpretation of the institutional dimension of actions... In the following we give more details about the default behaviour specified by Institutional Capabilities.

4. GENERALITIES ABOUT INSTITUTIONAL AGENTS

4.1 Types of agents

JIA allows to design three basic types of agents.

The **institution agent** (unique in each institution) is the agent managing the registry of the institution, thus omniscient about the institutional facts and rules of his institution. It also "spies" (*i.e.* observes) all institutional actions performed by other agents, in order to be aware of the evolution of the registry of the institution (actions can modify this registry).

The **mediator** (also unique in each institution) has a database of debugging strategies to manage problems encountered by other agents. The mediator can be either reactive (only managing problems signalled by other agents) or proactive (autonomously detecting problems and trying to manage them before they are signalled).

Standard agents constitute the normal members of the institution. Their default behaviour can be customized by several parameters:

- laziness: a lazy agent only tries to fulfill his obligations once they have been explicitly notified to him by another agent;

- conscientiousness: a conscientious agent checks if other agents fulfill their obligations;

- trust: a trustless agent asks the institution for confirmation anytime he receives an information from another agent.

These parameters influence the initialisation of the agent's Institutional Capabilities, in particular his set of Semantic Interpretation Principles.

The behaviour of all these kinds of agents is specified by the content of their Institutional Capabilities, that is detailed in the following paragraphs: a default list of *SIPs* (paragraph 4.2) and a default set of *KBase* filters (paragraph 4.3).

4.2 Interpretation of Semantic Representations: list of SIPs

In Institutional Agents as in semantic ones, incoming Semantic Representations are interpreted by a list of Semantic Interpretation Principles before being possibly asserted into the agent's *KBase* (see figure 2). We list here the *SIPs* that are specific to Institutional Agents, sorted into several categories. We only briefly describe them in this paragraph, since the most important ones will be described in full details in the following sections.

SIPs in the *action* category interpret the specification or performance of actions.

- Institutional Action Declaration: it interprets the institutional specification of actions provided in the agent's configuration files, and installs the mechanisms to manage these actions (see paragraph 5.3).
- Institutional Action Done: it interprets the performance of any action in order to deduce its institutional effects and to inform concerned agents (see paragraph 5.4).
- Obliged Action Done: it is also triggered by the performance of an action, and retracts possible obligations fulfilled by this performance.

SIPs in the *interaction* category specify the agent's behaviour towards other agents, speech acts received from them, and the resulting commitments, beliefs, intentions or obligations.

- Commitment Interpretation: it interprets new commitments taken by other standard agents, and checks their validity (searches possible contradictions with previously existing commitments).
- Conditional Obligation Interpretation: it interprets conditional obligations, *i.e.* obligations that will hold once a condition becomes true; it thus adds an observer on the agent's *KBase* to monitor the condition in order for the agent to be aware of his obligation as soon as it becomes true.
- Grounded Belief Transfer: this SIP is only provided to trustless agents and make them control each information from another agent, by asking the institution if this agent has no contrary commitment on this information; the incoming belief is transferred only after a positive answer from the institution.
- Institutional Belief Transfer: it prevents agents from directly adopting beliefs of other agents if they are about institutional facts; these institutional facts are checked by asking the institution, and only transferred after its confirmation.
- Institutional Intention Transfer: it interprets intentions of other agents (resulting from a request from these agents for example) and decides to adopt them or not. Requests from the institution are always obeyed, while the adoption of intentions from standard agents depends on these agents' institutional powers.
- Obligation Interpretation: it interprets new obligations for another agent to perform an action, and settles the monitoring mechanism in cases where the interpreting agent should monitor the respect of this obligation (see paragraph 6.1.1 for details).
- Obligation Notification: it interprets the notification of one of the agent's existing obligations by another agent, and forces the reinterpretation of this obligation in order to possibly trigger the behaviour to respect it (indeed lazy agents only try to fulfill their obligations once they have been explicitly notified).

SIPs in the *generic* category implement some axioms of the logic and some useful generic mechanisms.

- Since Formula Interpretation manages the predicate since, describing formulas that become true after another formula has become true. This is done with an observer that monitors the condition formula in order to interpret the main formula when it becomes true.
- Until Formula Interpretation manages the predicate until used to describe formulas that are only true until another formula becomes true. This is done with an observer monitoring the condition formula in order to retract the main formula when it becomes false.
- Period Formula Interpretation interprets the predicate *period* as an abbreviation defined in function of the *until* and *since* predicates.
- Institutional Since Formula Interpretation and Institutional Until Formula Interpretation implement mix axioms defining the links between temporal operators (since and until) and institutional ones.
- Split Institutional Fact: it implements the axioms defining the distribution of institutional fact operator over the *and* operator.
- Time Predicate: it allows agents to manage quantitative periods of time (for example wait for 5 seconds before performing an action).

The *planning* category gathers *SIPs* that modify the agent's planning.

- Future Obligation Interpretation: this allows agent to immediately interpret their future obligations (that will become true in the future, for instance encapsulated in a *since* formula) in order to start trying to fulfill them at once.
- Obligation Creation: it checks if the agent's goal can become an obligation for some agent by exerting some power. If so, the interpreting agent performs the procedure of this institutional power, what obliges the other agent to perform the intended action. For example a parent who has the intention that his son's room be clean has the power to command the son to clean the room.

- Obligation Transfer: this SIP defines a basic obeying behaviour for Institutional Agents, *i.e.* these agents always adopt the intention to fulfill their obligations. This SIP can be customized to more finely define conditions of obedience.

- Perseverance: this SIP makes the agent persevere when he fails to perform an obliged action. Actually he will monitor the feasibility precondition of this action and try again to perform it once it becomes feasible.

The *mediation* category gathers *SIPs* that are not provided to standard agents but only to the institution and/or mediator, to allow them to manage problems in their institution (inter-blocking obligations, violated obligations, complaints from standard agents).

- Blockade Detection: this *SIP* is used only by the mediator, and detects when a new obligation provokes an inter-blocking with an existing one. An inter-blocking is a situation where two obliged actions are both impossible to perform unless the other one is performed first. For instance when a client sends a purchase order to a provider, he gets obliged to pay and the provider gets obliged to deliver; but it may be impossible for the client to pay before delivery, and impossible for the provider to deliver before payment.

- Complaint Managing: this *SIP* allows the mediator and the institution agent to handle complaints from agents about violated obligations in their institution. The institution agent delegates to the mediator, and the mediator starts monitoring the obligation (see paragraph 6.2 for details about the management of violations by the mediator).

- Mediation Failed: this *SIP* allows the institution agent to react when the mediator informs him that he was not able to resolve a problem (a violated obligation). In this case the institution agent looks for the appropriate sanction in a specification file and applies it to the guilty agent.

4.3 Knowledge Base management: list of filters

The Institutional agents' KBase is provided with a set of filters responsible from the specific assertion and queries of institutional formulas.

- CountAs filter: count as formulas (normative consequence) are stored in nested KBases, each one corresponding to one institution (an agent can belong to several institutions); they are then queried from the appropriate nested KBase.

- Institutional facts filter: it handles a table associating each institution with a list of formulas that are true in the context of this institution (its "registry"). The assertion filter asserts institutional fact in the registry of the corresponding institution, while the query filter seeks institutional facts in the correct registry. An additional query filter is used if the queried fact is not directly stored in the agent's KBase: this filter scans the count as rules to find one allowing to deduce this fact, and then queries its premise; this allows to manage several steps of institutional deductions.

- Institutional omniscience filters: this query filter is only added to the institution agent's KBase. It implements the fact that if an institutional fact cannot be retrieved in the corresponding institution agent's KBase, then it is false.

5. INSTITUTIONAL INTERPRETATION OF ACTIONS

5.1 From intentions to action in JSA agents

As an introduction, we remind here the mechanisms provided by JSA to make agents behave depending on their intentions. We have not modified this mechanism but it will be useful to understand the sequel of the paper.

When an agent interprets one of his own intentions, it first goes through the Goal Commitment SIP that checks if the agent already has this intention, or if he has already reached it. In both cases, the intention is consumed by this SIP, otherwise it is propagated to the next SIPs.

Rationality Principle and Action Performance are the two basic planning SIPs available for JSA agents (the developer can add new ones to allow agents to perform a more sophisticated planning).

- Rationality Principle interprets an intention that some proposition holds, and checks all available actions (in the agent's table of semantic actions) to find one whose effect matches this proposition. If there exist one such action this SIP returns the corresponding plan, otherwise the intention is considered to be unreachable.

- Action Performance interprets an intention that some action is performed the agent, and returns the plan consisting in performing this action.

Now in the following subsections we detail what is new in JIA to manage the institutional dimension of actions performed by agents.

5.2 Specification file

Institutional features of actions are specified in xSL (extended Semantic Language, the language described above in paragraph 3.3.1) in a configuration file "<name>.actions". Each action is characterised by five features:

- the name of the concerned institution;

- the pattern of action;

- a list of agents that are spectators of this action (and should thus be informed of every performance of this action);

- a institutional precondition

- an institutional effect

This matches the characterisation of institutional actions provided in [2].

5.3 Interpretation of the action declaration

At the agent setup, the specification of each action is interpreted. In particular it goes through the *Institutional Action Declaration* SIP. This SIP performs four actions:

- it stores in the agent's KBase formulas allowing to retrieve the names of the observing agents, so that the author of an action knows which agents he has to inform of the performance of this action;
- if the interpreting agent is the institution agent or the mediator of the institution in which this action is declared to be institutional, it stores a formula to remember that the action belongs to this institution;
- it interprets the generic power for any agent to commit on the institutional precondition of this action by performing it (implicit effect in accordance with the semantic of institutional actions provided by [2]).
- it also interprets the powers corresponding to the deduction of the institutional effect specified for this action.

Once stored in the agent's KBase, these powers will be referred to by *Institutional Action Done* SIP when this action is performed, in order to automatically deduce their institutional implicit and explicit effect.

5.4 Interpretation of an action performance

Whenever an institutional action is performed, various agents are aware of its performance: its author, the observing agents who are informed by the author, and the institution agent and mediator of the corresponding institution, who spy all actions in their institution. These agents interpret the information about the performance of this action, in particular with the *Institutional Action Done* SIP.

If this action is the procedure of a power, this SIP checks its condition, and if it is true it asserts its institutional effect (both were interpreted from the actions specification file). If the agent performing this power procedure is conscientious (see definition in paragraph 4.1) this SIP also makes him monitor the fulfilment of the obligations he possibly created.

In the special case where the interpreting agent is the author of the action, this SIP makes him inform other relevant agents of its performance:

- observing agents specified in the action declaration;
- interested agents, who have manifested this interest by requesting the performance of this action or by notifying an obligation to perform it;
- concerned agents, *i.e.* agents who have new obligations created by the performance of this action;
- the institution agents and mediators of all institutions he belongs to (these agents select relevant information thanks to the list of actions belonging to their institution that they have constituted while interpreting the actions declarations).

6. LIFE CYCLE OF AN OBLIGATION

Obligations, among other institutional facts, are thus deduced by *Institutional Action Done* SIP from the performance of institutional actions. In this section we give more details about their subsequent life cycle, *i.e.* when agents decide to fulfill them, and when they are finally retracted. But we also give details about the management of violation cases: what happens when an agent cannot fulfill his obligations, how it is detected by other agents, how they can complain to the institution agent or the mediator, what these special agents can do to solve the problem, and which sanctions can be applied if no solving strategy is found.

6.1 Interpretation and monitoring of other agents' obligations

Standard agents are equipped with some interpretation rules to manage other agents' obligations.

6.1.1 New obligations

The *Obligation Interpretation* SIP is triggered when an agent observes a new obligation for another agent to perform an action. Two cases are managed:

- the interpreting agent is the mediator of the institution in which this obligation holds, he is proactive (definition in paragraph 4.1), and he is not already managing a blockade involving this action;
- or the interpreting agent is observing this action, and he is conscientious

In both cases an *Obligation Respect Observer* is added to the interpreting agent's KBase in order to monitor the performance of the obliged action, and to specify the watching agent's subsequent behaviour in case of violation.

6.1.2 Monitoring of obligations

An instance of *Obligation Respect Observer* is dynamically added to an agent's KBase when he wants to monitor another agent's obligation. If the watched obligation is fulfilled, the agent stops watching it, *i.e.* this observer is removed from his KBase. Otherwise a specific behaviour is triggered to react to the violation of the watched obligation.

The developer can specify various parameters of this observer to customize this behaviour:

- the number of notifications of the violated obligation to the responsible agent before complaining to the mediator or institution agent;
- the time interval between two successive notifications;
- the name of the mediator handling complaints about this obligation.

The violation management behaviour performed by standard agents consists in two steps. First, the watching agent notifies the violated obligation to the responsible agent; the number of notifications and the delay between them is specified by the developer. Second the watching agent complains to the mediator, or if no mediator is specified, complains directly to the institution agent.

The next section is dedicated to the different exception cases that can be encountered. We will in particular detail the violation management behaviour specified by *Obligation Respect Observer* for the mediator. We will also describe the managing of complaints sent by standard agents.

6.2 Mechanisms for exceptions

In this section we expose the various exceptions that can be encountered, and describe how the institution agent and the mediator can detect and manage them.

6.2.1 *Mediating strategies*

Agents observing a violated obligation can complain to the mediator. Proactive mediators are also able to detect violated obligations by themselves, from the spying of the agents' actions. Both means of detection lead to the same management behaviour, specified by *Obligation Respect Observer*. There are two cases.

If there is an inter-blocking, *i.e.* a situation where two obligatory actions are each one unfeasible while the other one was not performed, the mediator refers to a specification file (in xSL) listing predefined solving strategies. If there is no strategy, or if the strategy fails, the mediator informs the institution.

Otherwise, the mediator begins with notifying the violated obligation to the responsible agent, as do standard agents. But at the end of the final timeout, if the obligation was not fulfilled yet, he identifies the type of violation, depending on if the agent was trying to perform an unfeasible action, or if he was not even trying. He then signals the violation to the institution with a specific message that standard agents cannot send.

6.2.2 *Sanctions*

When the institution agent is informed by its mediator of an unsolved problem (mediation failed on an inter-blocking situation, voluntary or involuntary violation of an obligation), he refers to a configuration file listing pre-specified sanctions associated with various situations. It selects the appropriate sanction depending on the description of the situation made by the mediator, and declares this sanction to the guilty agent (for example the obligation to pay a fine).

7. CONCLUSION

In this paper we have proposed an extension for JSA allowing to develop Institutional Agents in the JADE platform. These agents are able to understand the semantics of FIPA speech acts, but also to reason about new concepts of norms, roles, powers... and to behave accordingly. Such institutional agents can thus evolve in an institutional context while respecting its rules.

This development framework has several assets. First it is compliant with the FIPA standard, that is widely used in Multi-Agent Systems. Moreover it blends mental attitudes (beliefs, intentions) with social ones (obligations, powers) allowing an expressive specification of the agents' behaviour. Second it grounds on a logical framework for institutional notions, ensuring the correctness of the reasoning of agents. This development framework will be made available open source soon.

Finally, designing agents able to deal with institutions and norms is essential for future applications of Multi-Agent Systems, notably those oriented towards electronic commerce. Besides we have experimented our JIA framework in an industrial application prototype. This project is still ongoing, and offers to businesses a flexible and evolutive solution to their interoperability problems.

8. ADDITIONAL AUTHORS

9. REFERENCES

[1] Anonymous. The reference will be made available in final version. In *???*, 2008. demonstration.

[2] Anonymous. The reference will be made available in final version. In *???*, 2008.

[3] R. Demolombe and V. Louis. Norms, institutional power and roles: towards a logical framework. In *16th International Symposium on Methodologies for Intelligent Systems (ISMIS'06)*, volume LNAI 4203, pages 514–523. Springer, 2006.

[4] FIPA. The foundation for intelligent physical agents. `http://www.fipa.org`.

[5] JADE. The java agent development framework. `http://jade.tilab.com`.

[6] V. Louis and T. Martinez. An operational model for the FIPA-ACL semantics. In *AAMAS'05 workshop on Agent communication (AC'05)*, 2005.

[7] V. Louis and T. Martinez. *Developping multi-agent systems with JADE*, chapter JADE semantics framework. John Wiley and sons inc., March 2007.