Proceedings of the ATOP Workshop at AAMAS 2009

Agent-based Technologies and applications for enterperise interOPerability

AAMAS Workshop, 12 May 2009

Editors:

Klaus Fischer

Jörg P. Müller

James Odell

Arne J. Berre

Preface

Today's enterprises must adapt their software processes to work in open settings, such as online marketplaces and, more generally, the Web, where business relationships exhibit a high degree of dynamism. Moreover, open settings are characterized by the autonomy and heterogeneity of the enterprises. In such settings, interoperability is a key concern: how do we ensure that diverse enterprises can work together toward a mutually desirable end? Interoperability problems occur at different levels: at the business level (how organizations do business together, what needs to be described and how?), at the knowledge level (different formats, schemas, and ontologies), and at the infrastructure level (the underlying information and communication technologies and systems). Agents, Model-Driven Architecture (MDA), and Service-Oriented Architecture (SOA) are complementary approaches to addressing the enterprise interoperability problem.

Agent technologies provide a cross-cutting approach promising to enable intelligent and proactive automation, adaptive planning and execution, decentralized coordination, and semantic interoperability. Agents enable dynamic collaboration and orchestration in changing and unpredictable situations. MDA supports interoperability due to its promise of providing consistent models at different abstraction layers with well-defined mappings in between these layers and provides mechanisms that generate artifacts for different platforms. SOA tries to reach interoperability, focusing upon, but not restricted to, the information and communication technology (ICT) level. It provides late-binding interoperability between business process requirements and providers of service implementations which results in loose coupling among software entities representing business objects (processes, organizational units, etc.).

The workshop focuses on technologies that support interoperability in networked organizations, on successful applications of these technologies, and on lessons learned. The main goal is to stimulate a discussion on in how far agent technologies can support interoperability in this context and to compare current trends in the development of agent technologies with recent developments in service-oriented and model-driven system design with respect to their ability to solve interoperability problems. Regarding model-driven system design the presentation and discussion of metamodels of the underlying technologies like for example agent technologies and service-oriented architectures is especially of interest.

Klaus Fischer
Jörg P. Müller
James Odell
Arne J. Berre

Budapest, May 2009

Organization

Klaus Fischer	DFKI, Germany
Jörg P. Müller	Siemens AG, Germany
James Odell	Oslo Software, USA
Arne J. Berre	SINTEF, Norway

Program Commitee

Sahin Albayrak	TU Berlin, Germany
Bernhard Bauer	University Augsburg, Germany
Amit Chopra	North Carolina State University, USA
Michael Georgeff	Monash University, Australia
Dominic Greenwood,	Whitestein Technologies, Switzerland
Axel Hahn	University Oldenburg, Germany
Christian Hahn	DFKI, Germany
Øystein Haugen	SINTEF, Norway
Sebastian Kämper,	IWi, Germany
Stefan Kirn	Hohenheim University, Germany
Margaret Lyell	IAI, USA
Saber Mansour	Oslo Software, France
Nikolay Mehandjiev,	Manchester Business School, UK
Michele Missikoff	LEKS, Italy
Eugenio Oliveira	University of Porto, Portugal
Herve Panetto	University Nancy, France
Omer Rana	Cardiff University, UK
Ralph Ronnquist	Intendico Pty. Ltd., Australia
Rainer Ruggaber	SAP, Germany
Omeir Shafiq	Digital Enterprise Research Institute, Austria
Iain Stalker,	Unversity of Teesside, UK
Ingo Timm	Goethe-University Frankfurt/Main, Germany
Jörg Ziemann	IWi, Germany
Ingo Zinnikus	DFKI, Germany

Table of Contents

A multi-agent mediation platform for automated exchanges between businesses

Carole Adam[1], Vincent Louis, Fabrice Bourge[2], and Sebastien Picant[2]

[1] RMIT University, Melbourne VIC 3000, Australia,
carole.adam@rmit.edu.au
[2] Orange Labs, Caen, France,
firstname.surname@orange-ftgroup.com

Abstract. To automate electronic exchanges between business, the classical approach is to define beforehand an interacting protocol that must then be rigorously followed. This imposes a costly design time and a constrained runtime. We thus adopt a different approach, representing companies with autonomous agents whose interaction is mediated by an additional agent able to anticipate and resolve interoperability problems at runtime. We build these agents using the agent platform JADE and more precisely we designed an institutional plugin for JADE called JIA, allowing agents to reason about institutional concepts such as obligations, norms and powers. This paper describes the functioning of JIA.

1 Introduction

To automate electronic exchanges between business, the classical approach is to define beforehand an interacting protocol that must then be rigorously followed. This imposes a costly design time and a constrained runtime. Several works aimed at proposing standards for frequently exchanged documents [12, 5] and proposing libraries of reusable data models [11, 10], thus reducing design time, but obliging businesses to adapt their processes to these standards. We thus adopt a different approach, representing companies with autonomous agents whose interaction is mediated by another agent able to anticipate and resolve interoperability problems at runtime. We build these agents using the agent platform JADE (Java Agent DEvelopment Framework, [7]) and its extension JSA (JADE Semantics Add-on, [9]) allowing to develop cognitive agents in compliance with the FIPA-ACL formal specifications [6]. More precisely we designed an institutional extension for JSA called JIA, allowing agents to reason about institutional concepts as formalised in Demolombe and Louis' logic of norms, roles and institutional powers [4]. This paper describes the functioning of JIA and of the mediation platform showing an example of the kind of applications that the JIA framework allows to achieve[3].

[3] For now the mediation platform is a prototype used to demonstrate the possibilities offered by the JIA framework. It has not been practically tested yet.

2 The mediation platform

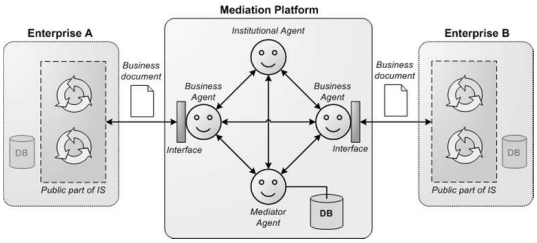

Fig. 1. Architecture of the B2B mediation platform

Each enterprise has its own business process constraining its interaction with other enterprises, but interoperability problems may arise if the business processes of two interacting businesses do not match. Besides it would be too costly to redefine the business process each time the enterprise has to interact with a new partner. Now we can consider B2B exchanges as 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 offer a flexible solution to the potential interoperability problems.

We thus designed a multi-agent B2B mediation platform [1, 3] 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; and 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.

In an example 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. This is where our mediator agent can intervene by finding in its library a debugging strategies 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 Functioning of JSA and JIA

3.1 JSA

Principle The JADE Semantic Add-on (JSA, [9, 8]) is a rule-based motor compliant with the BDI logic used to formalise the semantics of the speech acts of the standard FIPA-ACL [6]. 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, provided that they use the same ontology. 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.

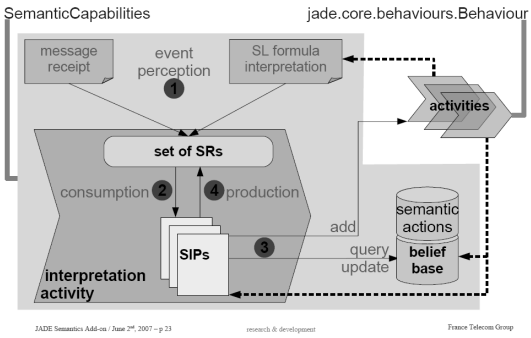

Fig. 2. JSA interpretation engine

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.

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. Its output is 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.

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.

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.

3.2 JIA

JIA is 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 3.2). The code allowing agents to deal with these rules is automatically generated.

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 [4]. This extended language is called xSL and features in particular operators for:

- 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.

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 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 and managing problems before they are signalled).

Standard agents constitute the normal members of the institution. Their default behaviour can be customized by various parameters: laziness (do not fulfill its obligation until explicit notification); conscientiousness (monitors other agents' obligations); and trustlessness (asks the institution to confirm any received information).

The behaviour of all these kinds of agents is specified by the content of their Institutional Capabilities. First their KBase is provided with a set of filters responsible from the specific assertion and queries of institutional formulas; we do not have enough space to list them here. Second they have a default list of *SIPs* that is detailed in section 4.2.

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: it 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: prevents agents from directly adopting beliefs of other agents 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 (*e.g.* resulting from a request from these agents) and decides to adopt them or not. Requests from the institution are always obeyed, while the adoption of intentions from standard agents depends on their institutional powers.
– Obligation Interpretation: it interprets new obligations for another agent to perform an action, and settles the monitoring mechanism if the interpreting agent should monitor the respect of this obligation (see parag. 6.1 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 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: 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 this son to clean it.
– Obligation Transfer: 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.

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 3.2) 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; and an institutional effect. This matches the characterisation of institutional actions provided in [1].

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 that 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 [1]).
– 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.

New obligations The *Obligation Interpretation* SIP is triggered when an agent observes a new obligation for another agent to perform an action, in two cases: either 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.

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

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.

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.

In [2] the authors propose a methodology for 3D electronic institutions aiming at integrating humans and agents in a virtual 3D world. The institution is specified and validated at design time so that no blocking can occur at runtime, while our approach consists in detecting a blocking in real time and mediating between the involved agents to resolve it. Moreover in this framework the participants have to follow well-defined interaction protocols, which is more rigid than the interaction allowed by the use of FIPA speech acts in our framework.

Our development framework[4] has several assets. First it is compliant with the FIPA standard, that is widely used in MAS. 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 agents' reasoning.

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 applied our JIA framework to the design of an industrial application prototype, offering to businesses a flexible and evolutive solution to their interoperability problems.

References

1. C. Adam, V. Louis, and R. Demolombe. Formalising the institutional interpretation of actions in an extended BDI logics. In *ESAW*, 2008.
2. A. Bogdanovych, M. Esteva, S. Simoff, C. Sierra, and H. Berger. A methodology for 3D electronic institutions. In *AAMAS*. ACM, 2007. poster.
3. F. Bourge, S. Picant, C. Adam, and V. Louis. A multi-agent mediation platform for automatic b2b exchanges. In *ESAW*, 2008. demonstration.
4. R. Demolombe and V. Louis. Norms, institutional power and roles: towards a logical framework. In *ISMIS*, volume LNAI 4203, pages 514–523. Springer, 2006.
5. ebXML. Electronic business XML - iso 15000 standard. `http://www.ebxml.org/`.
6. FIPA. The foundation for intelligent physical agents. `http://www.fipa.org`.
7. JADE. The java agent development framework. `http://jade.tilab.com`.
8. V. Louis and T. Martinez. An operational model for the FIPA-ACL semantics. In *AAMAS'05 workshop on Agent communication (AC'05)*, 2005.
9. V. Louis and T. Martinez. *Developping multi-agent systems with JADE*, chapter JADE semantics framework. John Wiley and sons inc., March 2007.
10. UBL - OASIS. Universal Business Language Technical Committee. `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ubl`.
11. UN/CCL. United nations core component library. `www.unece.org/cefact/codesfortrade/codes_index.htm`.
12. UN/EDIFACT. United Nations Directories for Electronic Data Interchange for Administration, Commerce and Transport. `http://www.unece.org/trade/untdid/`.

[4] This framework is now available open source as a community add-on for JADE [7].

Integration of agent-based Manufacturing Execution Systems into a Service Oriented Architecture

Sven Jacobi[1], Christian Hahn[2], David Raber[2]

[1] Saarstahl AG
Hofstattstrasse 106, 66333 Voelklingen, Germany
Sven.Jacobi@saarstahl.com

[2] German Research Center for Artificial Intelligence
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
{Christian.Hahn, David.Raber}@dfki.de

Abstract. The production of steel normally constitutes the inception of many supply chains in different areas of industry. Therefore, steel manufacturing companies are strongly affected by bull whip effects and other unpredictable influences along their production chains. In the course of these integrated operations, making the right decision at a certain stage can be the difference between earning or losing a great benefit. Improving their operational efficiency is required to keep a competitive position on the market. Hence, flexible planning and scheduling systems are needed to support these processes, which are based on considerable amounts of data, hardly processable manually anymore. MasDISPO_xt is an agent-based generic online planning and scheduling system for the observation on MES-level of the complete supply chain of Saarstahl AG, a globally respected steel manufacturer. This paper concentrates on the horizontal and vertical integration of influences of rough planning on detailed and the other way around. Based on model-driven engineering business processes are modeled on CIM-level, a service oriented architecture is presented for the interoperability of all components, legacy systems and others wrapped behind services. Based on this, an agent-based detailed planning and scheduling ensuring interoperability in horizontal and vertical direction is approached effectively.

1 INTRODUCTION

The production chain of Saarstahl AG consists of a multitude of specialised and complex metallurgical manufacturing processes with a lot of dependencies among them. First, a blast furnace factory produces hot metal from iron ore, coke and limestone as raw materials for the steel production. In fixed intervals distributed over the day, a determined quantity of hot metal is sent by rail to the steel works for the next production step. Inside the melting shop, steel of different quality grades is produced, according to concrete customer orders and requirements. It is cast at continuous casting plants into billets. A single production unit inside the steel works is called heat. A heat is part of a sequence – a total ordered set of similar qualities of equal formats.

Afterwards, these billets are delivered to the rolling mills. Here, steel bars or wire rods of different shapes and formats are produced. In fixed, cyclic rolling campaigns

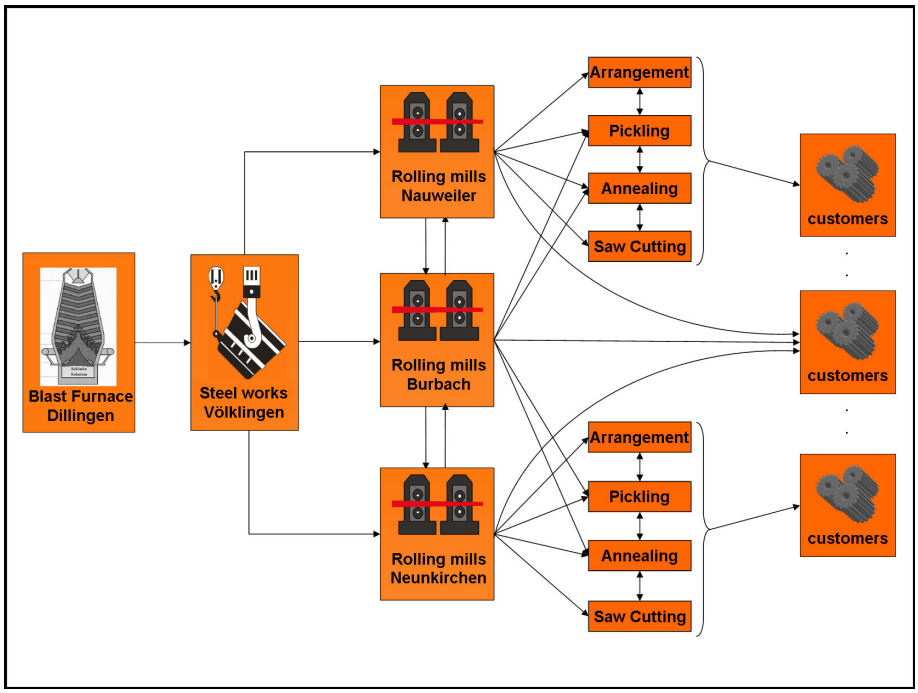

Fig. 1: Supply Chain of Saarstahl AG

of limited capacities certain formats are produced. These cycles are dependent on the rolling mills, billet supply by the steel works and concrete orders by customers, of course. They vary between one to four weeks. After rolling, follow-up manufacturing steps concerning steel bars are arrangement, pickling, annealing and saw cutting; wire rods probably need a annealing and a pickling. Finally, the products are delivered to the customers – mostly suppliers of the automotive, shipbuilding or aerospace sectors. Figure 1 depicts the roughly described Supply Chain.

Given a working plan, MasDISPO_xt schedules the execution of each concrete order along the production chain. It monitors production on a rough—in weeks—and detailed—in days and hours—level, and executes an online detailed planning and scheduling for the different manufacturing steps in each single factory. It has to detect problems in the production and handle them in order to return to normal production. The rough working plan for each manufacturing level (shown in Figure 2) is calculated on demand, before final order commitment. Depending on delivery date and order quantity certain capacities at specified aggregates have to be roughly allocated.

Usual orders to Saarstahl vary between five to several hundreds of tons. Batch sizes on each manufacturing level are fixed or limited, hence, orders have to be grouped together in process units on each stage with local constraints to keep. For instance, inside the steel works, a heat has a fixed size of $160t$. The orders covered by a heat have to be of same quality, same casting format and should have the same calculated processing step date. Additional restrictions concerning the production inside

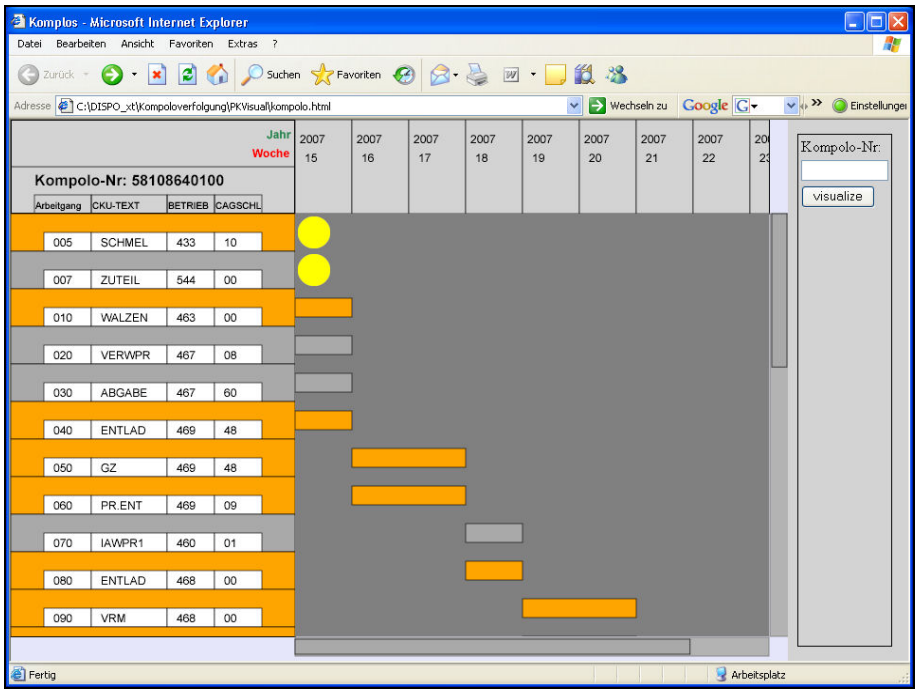

Fig. 2: Rough planning according to order position

the steelwork describe a local problem. This is decribed in detail in MasDISPO – the Multiagent System [Weiss, 1999], [Wooldridge, 2002] for the steelworks' optimisation [Jacobi et al., 2007].

It is not only the batch size which complicates mapping from one phase to another. Inside the steelworks, orders are also grouped together by steel grades, whereas, on the next phase – rolling – physical dimension is the most relevant criterion; while for annealing treatments, the temperature is the most important one. Thus, every single order has to be mapped into different units on different production phases. Therefore, the groups created by each of these criterion cause production dependencies among the orders and, consecuently, changes in one order's schedule may impact the other's schedules at a given phase.

The average order backlog at Saarstahl is about 17500 orders, which makes it already a complex challenge to find an optimal mapping which keeps all constraints and deadlines. However, the system has to, additionally, deal with the online problem of dealing with new incoming orders and changing requirements by customers.

Normally, as a process step gets closer to a certain phase, the more concrete its allocation and the more detailed its planning has to be. Therefore, this system has to deal with smooth transitions between rough and detailed planning – a challenge which is often only non satisfying matched by traditional centralised approaches [SAP, 2004]. Dependencies between rough and detailed planning, as well as, interconnections between different manufacturing phases have to be modeled.

As presented, the overall process chain is characterized by changes in customer orders and it is affected by production setbacks or problems. Therefore, steel manufacturing companies must be flexible and dynamic, by adapting production plans fast in order to meet customer requirements while still being cost-efficient.

Since these are requirements which need to be covered in almost every industrial sector, there are a lot of commercial systems handling this. But these ERP systems (enterprise resource planning) [Gronau, 2004] like APO (Advanced Planner and Optimizer) [Bartsch and Bickenbach, 2002] or APS (Advanced Planning and Scheduling) are suitable for a rough planning only, but are frequently not very suitable for operations planning. These existing solutions are dominated by centralized decision making processes, mostly data driven and often not modeling the business processes they should. Big software companies have adopted the strategy to provide integration mechanisms for MES-level solutions [SAP, 2004] like the presented solution.

MasDISPO_xt, a decentralised agent based approach, is the proposed solution of this paper. In MasDISPO_xt, every order is modeled as an agent. The agent calculates and observes its own schedule from order entry, across rough and detailed planning, and monitors the production up to the point of delivery. It responds to changes during planning, scheduling and production by dynamically adapting the schedules. Also, each aggregate of any factory is also modeled as an agent which also calculates its schedule autonomously based on further local knowledge and restrictions.

The complete production chain is very complex and could not be addressed with the appropriate detail in the context of only one paper. Therefore, this paper concentrates on the horizontal and vertical integration of solutions provided for detailed planning into a global perspective regarding rough planning and the other way round. As a presetting, rough planning influences detailed planning and vice versa a rearranged detailed planning might impact the rough layer – probably on several manufacturing levels. Existing ERP solutions cannot provide such integrations very suitable – the motivation for the presented approach.

Section 2 describes the normal course of action concerning planning at Saarstahl in more detail. Based on this, specific requirements are identified. Section 2 closes with the presented approach. A general discussion of the deployed solution is described in 3. Also, conclusions and ongoing work are presented.

2 Planning and Scheduling at Saarstahl AG

2.1 Problem description

Steel production in general is a very complex, dynamic and disassembling process. It starts inside the blast furnace as hot metal and ends up in vast number of products of different kinds. On each manufacturing step, there are lots of restrictions regarding quality, size, dimension and others which determine the campaigns and batch units on each processing step. Of course, time restrictions also have to be kept. Hence, a delay on a certain step might have cascading effects along different branches of further processing. As mentioned, the average order backlog of Saarstahl is about 17500 – each order composed of several order positions and each position with up to 50 processing steps. Hence, it is a challenge to keep planning and scheduling under control.

Competitive orientation concerning product concepts, technology and others are discussed per annum at Saarstahl and give a direction for human resource planning or machinery on a tactical level. In planning levels regarding sales, the global production capacities for the different production phases are booked. After that, the planning process continues by planning at lower levels. For each factory, the global planning level provides the lower level with a set of orders for a specified time horizon. These assigned orders are planned in more detail while going down in planning levels down to physical production systems. All these influences are correcting variables from 'above' often given just once.

On the other hand, physical production systems might have any delays which have to be adopted by the planning system above. Delivery reliability or throughput are measured constantly and influence the next level above. These control variables go up to the strategical planning on highest level. A cycle of interdependencies as shown in figure 3 is created. Flexibility across these different layers is needed to guarantee a flexible planning and production process.

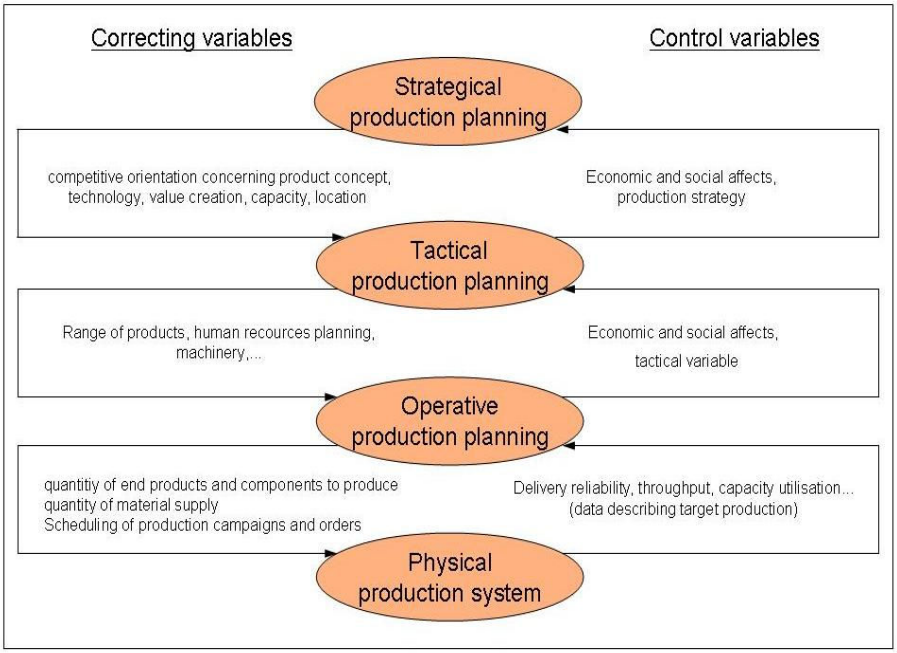

Fig. 3: Control Cycle of planning influences and dependencies

The planning horizon on highest level is quite rough, the closer it gets down to production the shorter the horizon is. To realise this exchange of information, smooth interfaces between the acting systems are needed. On a higher level planning level, usually normal ERP-systems (enterprise resource planning) are used. These systems probably also provide a advanced planning system (APS) for a rough planning. Beneath,

a so called manufacturing execution system is used for a detailed planning. The last level is the automation level covered by 'SPS' or 'SCADA' (Supervisory Control and Data Acquisition). A smooth vertical integration of these acting systems is needed to guarantee a fast exchange of informations which is especially needed in steel production since materials are often molten between two different processing steps and might not cool down. Figure 4 depicts again the control cycle but from a system's point of view.

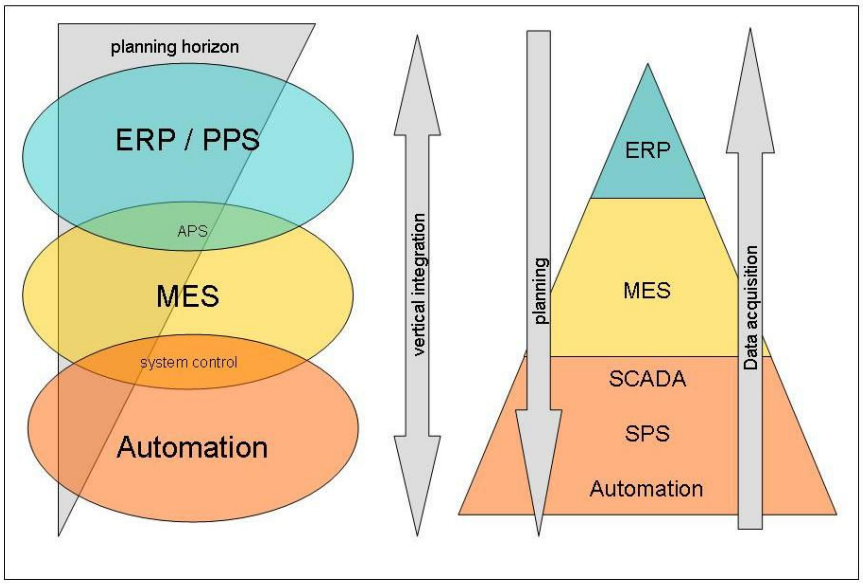

Fig. 4: Planning pyramid at Saarstahl AG

Saarstahl is a historically grown company. Hence, a lot of obsolete legacy systems are still running. Integrations as described above are especially regarding dynamic aspects viable unsatisfying by existing, commercial approaches. Driven by this need Saarstahl decided to develop own solutions and participate in research projects like the European funded project 'SHAPE' (Semantically-enabled Heterogeneous Service Architecture and Platforms Engineering) [SHAPE, 2008] which are dealing with these problems.

2.2 Solution

The main idea in order to be able handling these problems is to model each single order position as a software agent. Every single compolo agent (*com*ission - *po*sition - *lo*t) calculates and observates its own schedule from order entry until invoicing. Instead of handling a vast number of restrictions subject to the manufacturing step in general only a few which are relevant for a single order position are handled by the entity autonomously. A decentralised management of manufacturing control is received instead of a centralised, data driven approach.

Definition 1. *Having a finite set of* factories *F with elements f_i, where f_i is the factory number i,* aggregates *A with elements a_j, where a_j is the factory number j*

$$F = \{f_1, \ldots, f_n\}\ n \in \mathbb{N}$$
$$A = \{a_1, \ldots, a_m\}\ m \in \mathbb{N}$$

O being the finite set of all orders *to be planned, with elements identified with the letter o,*

$$O = \{o_1, \ldots, o_q\}\ q \in \mathbb{N}$$

L_i being the ordered list of elements of A which are the suitable aggretates for order o_i in order of preference*,*

$$L_i = \{a_x, \ldots, a_y\}$$

where

$$\mid L_i \mid \leq n;\ i = 1, \ldots, q;$$
$$a_x \in L_i \wedge a_y \in L_i \wedge$$
$$a_x \text{ precedes } a_y \text{ in the list } L_i \Rightarrow$$
$$a_x \text{ is preferable over } a_y \text{ for order } o_i$$

L being the collection of preferences for all orders:

$$L = \{L_1, \ldots, L_q\}$$

and the functions C, being the function which associates an aggregte to its associated available capacity for a given period, and c, the function which associates each order to its required capacity on an aggregate,

$$C : M \rightarrow \mathbb{N}$$
$$c : O \rightarrow \mathbb{N}$$

the top level plannning problem for assinging aggregates can be defined *as the search for a set $\mathcal{P}$ that associates each order in O to aggregates in A following the preferences provided by L and making sure that the sum of all sizes provided by $c(x)$ of the orders associated to a specific furnace do not exceed the furnace's specific maximal capacity $C(a_j)$:*

$$\mathcal{P} = O \times A$$

where

$$\forall a_j \in A : \left(\sum_{x \in \{o \mid (o,a_j) \in \mathcal{P}\}} c(x) \right) < C(a_j)\ .$$

A solution $\mathcal{P}$ is produced by searching for each order $o_i \in O$ (sorted by arrival date) a factory $f_i \in F$ and an aggregate $a_j \in A$ with available capacity, following the list L_i. Furthermore, S_{o_q} is the finite set of processing steps with elements s_k, where s_k is the step number k necessary to meet customer requirements according o_q. H is the function assigning the number of different factories along the process chain of each single order position.

$$S = \{s_1, \ldots, a_l\}\ l \in \mathbb{N}$$
$$H : S_{o_q} \times S_{o_q} \rightarrow \mathbb{N}$$

$$h(s_k, s_{k+1}) \mapsto \begin{cases} 0 : f(a_{s_k}) = f(a_{s_{k+1}}) \\ 1 : f(a_{s_k}) \neq f(a_{s_{k+1}}) \end{cases};$$

$$f(a_{s_k}) \text{ maps step } s_k \text{ on aggregate } a_{s_k} \text{ to factory } i \; min \left(\sum_{S_{oq}} \sum_{k} h(s_{s_k, s_{k+1}}) \right)$$

By minimizing this function transportation costs will be optimised. Of course, this is not the single objective. Along the complete process chain, there are several and also conflicting goals. But this is not addressed in this paper.

This general definition is valid across the complete supply chain for every single order position in each local subsystem. Additional restrictions and others are taken into account while going down on more detailed level. The degree of detailedness also defines the framework for information exchange. The goal is to exchange as least data as possible but still enough to guarantee transparency as demanded.

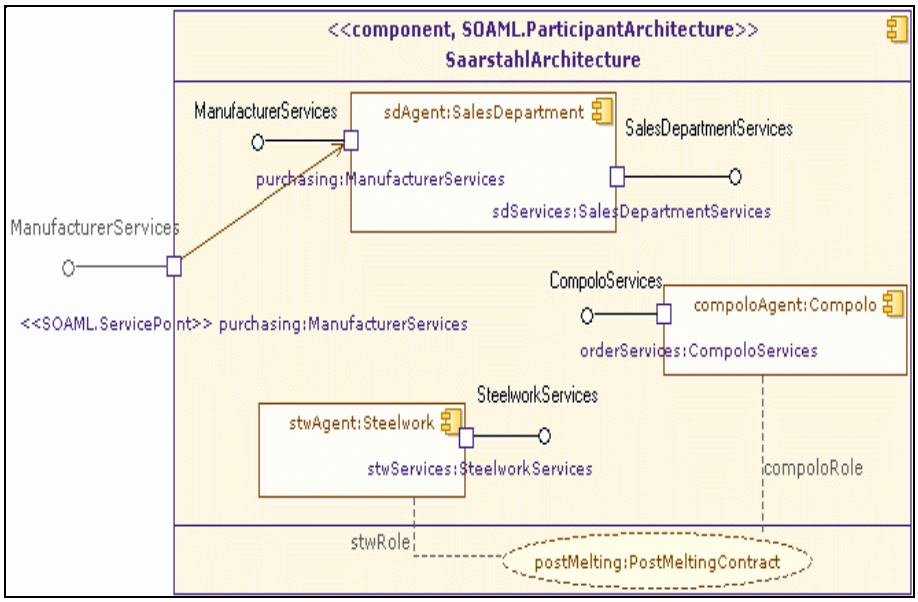

Fig. 5: Service Oriented Architecture of Saarstahl AG

How are existing legacy systems integrated into workflow and how are interactions realised? This is solved by use of a *Service Oriented Architecture* (SOA). The idea of the service oriented approach is straightforward. Existing legacy systems are wrapped behind services which are part of the service oriented architecture of Saarstahl. A simplified model of this architecture in which a lot of participants have been omitted is choosen to explain this approach in more detail. Figure 5 depicts this model which is based on SOAML [OMG, 2008]. In this scenario the internal architecture of Saarstahl consists of only three participants namely sales department, steel works and compolo agent. Each participant offers a number of services to wrap the functionality of the corresponding legacy systems in its domain. The behavior of each participant as well as points of interaction with other participants are modeled by UML activities. Figure 6 depicts a sample interaction showing the steel works participant sending a production report to the compolo agent.

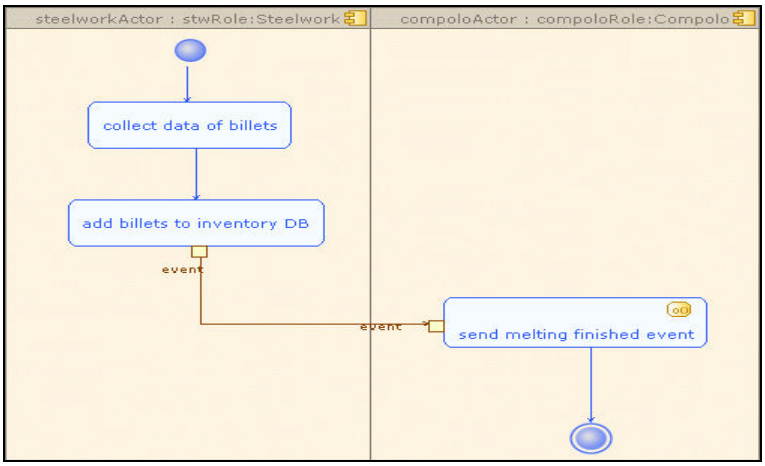

Fig. 6: Activity: Steelwork informs compolo agent

The SOAML model represents an intermediate model in the whole model driven development process only. The planning process and requirements are formalized for the first time on computation-independent model (CIM level) via business process modeling notation (BPMN) using ARIS [IDS Scheer AG , 2008] . The activity in figure 6 has been generated from the models while going down from CIM to PSM level. *Services* like *'Inform Service'* provided by the compolo agent are specified and assigned. A model transformation to PIM level (platform-independent model) is done using SOAML resulting in a model as shown in figure 5.

This is followed by another model transformation on PIM level to *PIM4Agents* [Hahn et al., 2007]. The metamodel of agent aspect is centered on the concept of an *agent*, the autonomous entity capable of acting in a specified environment. The transformation to *PIM4Agents* is realized using the *ATL* (ATLAS transformation language) [Eclipse-Foundation, 2009].

Two relevant mapping rules of the transformation are presented next to show how they are applied for the presented model mappings.

Model mapping 1:
Head: SOAML:ParticipantArchitecture → PIM4Agents:Organization
Body: Every ParticipantArchitecture in the SOA model is mapped to a Organization a specialisation of an agent. The details of this mapping are summarized by the following table.

Target	**Source**
Required roles	OwnedAttributes which are of type ParticipantArchitecture
Performed roles	Every occurrence as an OwnedAttribute in some ParticipantArchitecture or ServiceArchitecture
OrganizationUse	Owned CollaborationUses
Interaction	Corresponding ServiceContracts to owned CollaborationUses

In SOAML the concept ParticipantArchitecture is used to describe how internal participants work together for a purpose by providing and using services expressed as service contracts. This nicely corresponds to the notion of the Organization aspect in PIM4Agents which describes how single autonomous entities cooperate within the multi–agent system and how complex social structures can be defined. Collaborations which are contained inside an Organization define the cooperation. Figure 7 depicts the result of this transformation. An Organization is represented by a group of three actor symbols. The input is the architecture as depicted in figure 5.

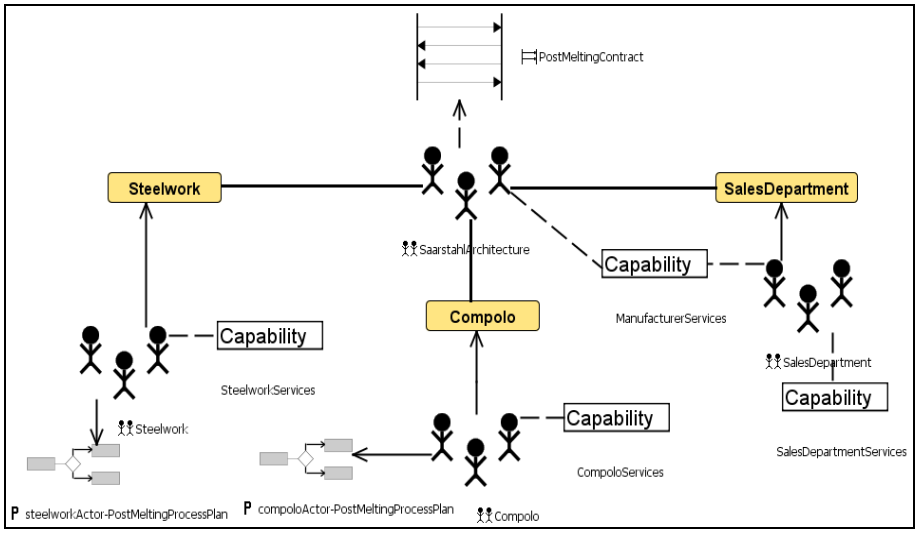

Fig. 7: Saarstahl Architecture modeled with PIM4Agents

Model mapping 2:
Head: UML:CollaborationUse → PIM4Agents:Collaboration

Body: The concept CollaborationUse in UML is mapped to a Collaboration that defines which organizational members are bound to which kind of Actor as part of an ActorBinding.

Target	Source
Organization	The containing ParticipantArchitecture or ServicesArchitecture
InteractionInstance	ServiceContract which types the CollaborationUse
Binding	Collection of DomainRoles which are required by the corresponding ServiceContract
actorBinding	Collection of roles bound to this CollaborationUse

This mapping is again a one to one correspondance but the Collaboration concept in PIM4Agents contains a bit more information than the CollaborationUse. Therefore information from both CollaborationUse and ServiceContract are required to fill a Collaboration.

The final transformation is going down to platform specific model (PSM). On this layer, a detailed planning – for instance concerning the production inside the steelwork as described in [Jacobi et al., 2007] – is fulfilled. Also other systems are involved. Figure 8 summarizes the transformations as described.

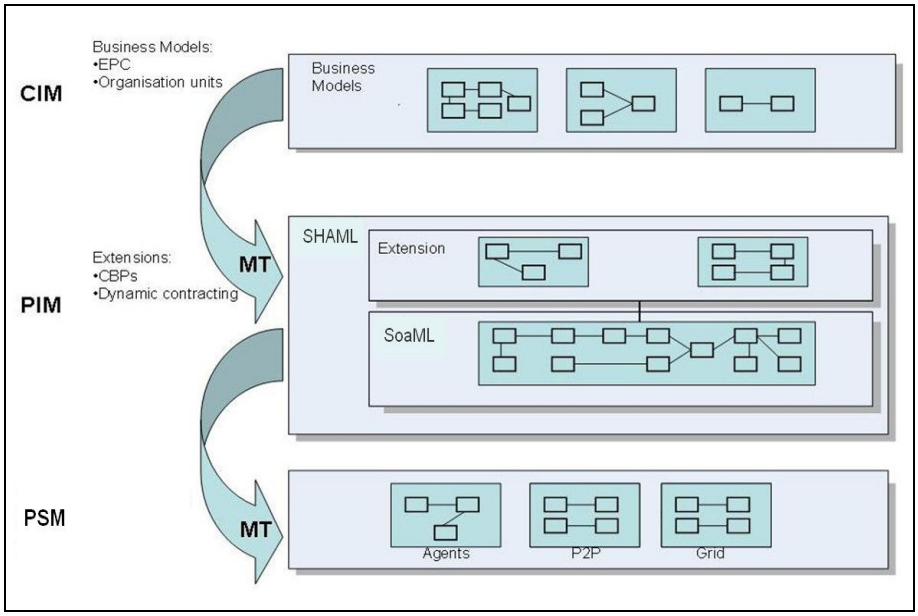

Fig. 8: Model transformations CIM-PIM-PSM

3 Application use and payoff & Conclusion

The need for flexible integrated planning systems with a clear workflow and a fast information exchange are undeniable. Existing, commercial approaches cannot guarantee such flexible interfaces. The agent based approach keeps calculation as 'local' as possible instead of calculating complete new schedules for entire processing units caused by any delay as provided by centralised planning tools. New solutions are related to old, but invalid ones. So, observers are able understanding how to get from one solution to another. By use of a SOA, vertical integration but also horizontal integration of information exchange is eased. Response times are reduced, because by use of services provided by orders or certain aggregates both modeled as agents, informations are exchanged without any detours anymore. No informations are lost, because services are embedded in specified protocols which ensure completeness.

The described examples of this paper state a subset of the different problems which need to be solved along production inside supply chains. An automatic and responsive planning system is needed. The decentralised approach with multiagent systems make the system easier to handle, really models the demanded business processes and is able to manage the huge data amount along production – requests which are not always met by the existing centralised approaches. By use of SOA, a more flexible environment easily to extent is received. The authors like to thank Saarstahl AG. Without their innovation related mind the realisation of this would not have been possible.

References

[Bartsch and Bickenbach, 2002] Bartsch, H. and Bickenbach, P. (2002). *Supply Chain Management mit SAP APO*. SAP Press, 2nd ed.

[Eclipse-Foundation, 2009] Eclipse-Foundation (2009). Atlas transformation language atl. http://www.eclipse.org/m2m/atl/.

[Gronau, 2004] Gronau, N. (2004). *Enterprise Resource Planning und Supply Chain Management: Architektur und Funktionen*. Oldenbourg (Muenchen).

[Hahn et al., 2007] Hahn, C., Madrigal-Mora, C., and Fischer, K. (2007). Interoperability through a platform-independent model for agents. *Proc. 3rd Inter. Conference on Interoperability for Enterprise Software and Applications (I-ESA 2007)*.

[IDS Scheer AG , 2008] IDS Scheer AG (2008). Aris. http://www.ids-scheer.com/en/ARIS/ARIS_Software/3730.html.

[Jacobi et al., 2007] Jacobi, S., Leon-Soto, E., Madrigal-Mora, C., and Fischer, K. (2007). Masdispo: A multiagent decision support system for steel production and control. *AAAI Innovative Applications of Artificial Intelligence*.

[OMG, 2008] OMG (2008). Service oriented architecture modeling language (soaml) - specification for the uml profile and metamodel for services (upms). http://www.omg.org/docs/ad/08-08-04.pdf.

[SAP, 2004] SAP (2004). Integration von mes-systemen in sap for mill products.

[SHAPE, 2008] SHAPE (2008). Semantically-enabled heterogeneous service architecture and platforms engineering. http://www.shape-project.eu.

[Weiss, 1999] Weiss, G., editor (1999). *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. KIT Press.

[Wooldridge, 2002] Wooldridge, M. (2002). *An Introduction to Multiagent Systems*. John Whiley & Sons.

A Modular Protocol Model for PIM4Agents

Esteban León-Soto,
Cristián Madrigal-Mora, Christian Hahn, Stefan Warwas and Klaus Fischer

DFKI GmbH
Campus D3.2,
D-66123 Saarbrücken, Germany
{Esteban.Leon, Cristian.Madrigal, Christian.Hahn, Stefan.Warwas,
Klaus.Fischer}@dfki.de

Abstract. For a long time, many models for interaction protocols have been proposed, very few of them approach the problem of modularity in interaction protocols. In the present work an implementation of an abstract model for interaction protocols is presented using a platform independent model for multi-agent systems (PIM4Agents). The advantages of using a declarative method to give semantics to actions and protocols come in handy at the time of representing in a concrete tool the correlation between actions. Having explicit semantics of what actions can do enables the possibility of having advanced tools that take advantage of it. The present work explains how the abstract protocol model is introduced in PIM4Agents, making it possible to develop model driven tools for interaction protocols in multi-agent systems.

1 Introduction

In multi-agent systems, communication between agents plays a crucial role. Since conversations can be quite complex and unpredictable, interaction protocols were invented, in order to reduce possible interactions to those necessary to accomplish a certain purpose. They are a key concept, since they are used to coordinate systems that are intended to work together in an interoperable manner. Great amounts of work have been invested into creating versatile and agile interaction protocol models, being those proposed by the Foundation for Physical Intelligent Agents (FIPA) the most popular [1]. Even so, none of these proposals provide a model that supports modularity and a concrete mechanism to rule how actions and protocols can be combined together to produce new protocols. Some of the proposals do address this aspect, but only "in spirit", since this task of combining is left completely for the designer to define and there are no concepts in the protocols that represent or support these decisions.

PIM4Agents is a metamodel that has been developed with the purpose of supporting model-driven development of multi-agent systems. Interaction protocols are one of the concepts present in this meta-model, among many others.

This work will describe, how a modular model for protocols is described in PIM4Agents. This model has as main objective to support modularity.

As it will be explained in further detail, this model will define actions in a declarative way, using propositions about the context of conversation. Doing this is of crucial importance, separating actions from their semantic meaning: preconditions and effects allow to compare and a differentiate in an explicit way which actions can be connected together and which actions can replace other actions.
First we describe previous work done in the area of Multi-Agent Systems (MAS) modeling, in Section 2 we briefly expose a metamodel for multi-agent systems called PIM4Agents and in Section 3 we describe the abstract model we provide for achieving modularity in interaction protocols. After that, in Section 4 the way this model is implemented for PIM4Agents is described, followed by Section 5, using an example, we explain how this tool can be used. Finally, in Section 6 we evaluate and compare results.

2 PIM4Agents

For designing MAS, a platform-independent domain-specific modeling language called DSML4MAS [2] has been developed. Like any language, DSML4MAS consists of an abstract syntax, formal semantics (see [3] for more details) and concrete syntax (see [4] for more details). However, in this paper we mainly focus on the abstract syntax.
The abstract syntax of DSML4MAS is defined by a platform independent metamodel for MAS called PIM4AGENTS defining the concepts and their relationships. The core of the PIM4AGENTS is structured into different viewpoints briefly discussed in the remainder of this section.

- *Multiagent view* contains the core building blocks for describing MAS. In particular, the agents situated in the MAS, the roles they play within collaborations, the kinds of behaviors for acting in a reactive and proactive manner, and the sorts of interactions needed for coordinating with other agents.
- *Agent view* defines how to model single autonomous entities, the capabilities they have to solve tasks and their roles they play within the MAS. Moreover, the agent view defines to which resources an agent has access to and which kind of behaviors it can use to solve tasks.
- *Organization view* defines how single autonomous agents are arranged to more complex organizations. Organizations in DSML4MAS can be either an autonomous acting entity like an agent, or simple groups that are formed to take advantage of the synergies of its members, resulting in an entity that enables products and processes that are not possible from any single individual.
- *Role view* covers the abstract representations of functional positions of autonomous entities within an organization or other social relationships. In general, a role in DSML4MAS can be considered as set of features defined over a collection of entities participating in a particular context. The features of a role can include (but not be limited to) activities, permissions, responsibilities, and protocols. A role is a part that is played by an entity and can as such be specified in interactive contexts like collaborations.

- *Interaction view* focuses on the exchange of messages between autonomous entities. Thereby, two opportunities are offered: (i) the exchange of messages is described from the internal perspective of each entity involved, or (ii) from a global perspective in terms of agent interaction protocols focusing on the global exchange of messages between entities.
- *Behavior view* describes the vocabulary available to describe the internal behavior of intelligent entities. The vocabulary can be defined in terms of combining simple actions to more complex control structures or plans that are used for achieving predefined objectives or goals.
- *Environment view* contains any kind of Resource (i.e. Object, Ontology, Service etc.) that is situated in the environment and can be accessed and used by Agents, Roles or Organizations to meet their objectives.
- *Deployment view* describes the run-time agent instances involved in the system and how these are assigned to the organization's roles.

To lay the foundation for further discussions on how to use DSML4MAS for modeling interactions, we focus on the interaction viewpoint in the remainder of this section. Key PIM4Agents concepts for the present work are illustrated in Fig. 1.

2.1 Interactions and Organizations in PIM4Agents

The interaction aspect of DSML4MAS defines in which manner agents, organizations or roles interact. A *Protocol* is considered as a special form of an *Interaction*. In the deployment view, furthermore, the system designer can specify how *Protocols* are used within *Organizations*. This is done through the concept of a *Collaboration* that defines which organizational members (which are of the type *AgentInstance*) are bound to which kind of *Actor* as part of an *ActorBinding*. Beside a static binding, the designer may want to bind the *AgentInstances* at run-time.

An interaction protocol as a pattern for conversation within a group of agents can be more easily described using generic placeholders like 'Initiator' or 'Participant' instead of describing the interaction between the particular agent instances taking part in the conversation. In DSML4MAS, this kind of interaction roles are called *Actors* that bind *AgentInstances* at design time or even at run-time. Furthermore, *Actors* require and provide certain *Capabilities* and *Resources* defined in the role view of PIM4Agents.

Messages are an essential mean for the communication between agents in MAS. In PIM4Agents, we distinguish between two sorts of messages, i.e. *Message* and *ACLMessage* which further includes the idea of *Performatives*. Messages have a content and may refer to an *Ontology* that can be used by the participating *Actors* to interpret the *Message* and its content.

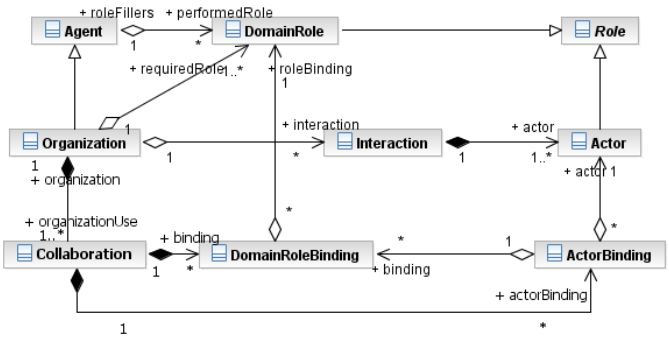

Fig. 1. Key Concepts in PIM4Agents.

3 Modeling modularity for Protocols

An interaction protocol model must have certain properties to be capable of supporting modularity. In this section these properties will be formally defined.

First of all, actions and their effects must be represented explicitly separated. In order to have a way of representing which actions and protocols (which are composed actions) can have the same effects, actions will be defined using some propositions that represent facts as seen from a global perspective.

Actions will be defined as operations over some propositions and a set of preconditions required by the action. These preconditions and operations will define the semantic of actions and will allow to replace or connect actions (or sets of them) with other ones that semantically have matching preconditions and effects.

A protocol is seen as a mechanism to control interactions. These are modeled as sets of possible actions and how these can be sequenced. Protocols will be defined as a transition system using a model similar to finite state machines. To represent how actions are to be organized: nodes are state of the complete system and actions are connections from one state to another.

In this specific case, state nodes will not be defined as a normal (concrete) state (where all propositions of the complete system and their truth value are specified), but as *state descriptions*, which represent sets of concrete states that fulfill some constraints. These constraints consist of true values assigned to some propositions, all concrete states belonging to the same state description will have the same truth value assigned to the propositions mentioned in the state description. Actions will be represented as a set of operations over some propositions, bringing about some facts or making them false. The post-conditions of an action can be calculated by applying its operations to the preconditions of the action.

Using state descriptions and *operations* cope with the Frame Problem [5] by allowing to refer only to what is relevant for each action.

3.1 Actions

Actions are a mapping of a state description, a label and a set of operations over some propositions to a state description.
A concrete state σ is an association of each proposition in P to a truth value ($true$ or $false$). A state description $s \in S$ is defined as the set of all concrete states σ that fulfill the constraints defined in s:

$$s(\langle p_a, true\rangle, \ldots, \langle p_k, true\rangle) = \{\sigma \in \Sigma | \langle p_a, true\rangle \in \sigma, \ldots, \langle p_k, true\rangle \in \sigma\} \tag{1}$$

where:
$k \leq |P|$
Σ: Set of all states.

The set of operations O is an association of propositions with an operator $+$ or $-$, meaning that the proposition p is set to true in case of $+p$ or to false in case of $-p$.
Actions must follow a *principle of effectiveness*, therefore the set of targeted states s' cannot be the same as the starting state description s. The set of all actions is called A:

$$\begin{aligned} &\Delta : S \times V \times \mathcal{P}(O) \times S \\ &A = \{\langle s, v, c \subseteq O, s'\rangle \in \Delta | (s' \neq s)\} \end{aligned} \tag{2}$$

where:
S: Set of all state descriptions
V: Set of all names for actions
O: Set of all operations over propositions in P of the form $+p$ or $-p$.
Participants of an action are represented as *roles* (r) members of the set of roles: $r \in R$. Every action is always performed by a role and is targeted at another role. The set A_R of actions associated to roles is:

$$A_R : R \times A \times R \tag{3}$$

where

$$a_i = \langle r_{xi}, \langle s_i, l_i, c_i\rangle, r_{yi}\rangle$$
$$\forall a_i \in A_R : r_{xi} \neq r_{yi}$$

3.2 Propositions

Propositions are the fundamental concept for modeling actions. All propositions p_i are members of the set of all distinct propositions P:

$$P = \{p_1, p_2, p_3, \ldots, p_n\}$$

Propositions can be of two kinds, regarding the way participants can be aware of their truth value: on one hand, perceptible propositions:

about facts that are observable by the participants, their truth value can be known using sensing capabilities, on the other hand non-perceptible propositions: about facts that can only be known if a participant takes part of an action which sets an explicit truth value for the proposition. There is no constraint on how propositions are to be defined and what they should represent. Their only concrete objective in the current model is to serve as mechanism to correlate similar actions. In the context of interaction protocols, there are some facts that require special observation, for which specific kinds of propositions will be defined:

Operational propositions Operational propositions are those that are defined as some arithmetic or logical statement:

- Arithmetic proposition: algebraic statement intended to define how facts are to be evaluated, for instance $bid > value$.
- Logical proposition: (also called conditional proposition) Used to express some fact that might be brought about if a logical term is valid, for instance $bid > value \Rightarrow won$ means, that in case bid is greater than $value$, won will be valid.

Timeouts There are certain cases where concrete time-windows are to be specified in an interaction protocol. For this purpose timeouts will be defined.
A Timeout $T(t_p, a) \in \mathcal{T}$, where $a \in A_R$ (the set of actions), $\mathcal{T} \subseteq P$, is a proposition member of the set of timeout propositions $\mathcal{T}$ that states that the action a will be performed after a certain period of time t_p that starts to count after the action that brings about the timeout is performed. Action a in a timeout is not necessarily performed by the sending role mentioned in a, but instead it can be an assumption the receiver of a can make. Also, this action will always have implicitly the operation $-T(t_p, a)$ declared, hence, timeouts are removed automatically after a is performed.

Commitments A commitment [6] $C(a_d, a_c, p, c, T) \in P$ is defined as the commitment of the *debtor* agent a_d to the *creditor* agent a_c to bring about the proposition $p \in P$ under the condition that the proposition $c \in P$ becomes true. After the condition c becomes true, agent a_d is expected by agent a_c to perform some action that produces p to be true. This action is to be performed before timeout T that represents the time limit is enabled. This timeout starts to count as soon as the condition c is brought about.
An *unconditional* Commitment $C(a_c, a_d, p, T)$ is an abbreviation of a commitment: $C(a_d, a_d, p, true, T)$ which simply means that agent a_c expects a_d to bring about the proposition p within the time period specified in T which is already running.

3.3 Interaction Protocols

An interaction protocol is a structure that represent a set of actions that are organized in a desired way. It serves a purpose of encapsulating these sets of actions in order to ease their management.

A protocol π is an association of preconditions in the form of one or more state descriptions, a label, post-conditions in the form of one or more state descriptions and a set of "roled" actions. The set of all protocols is called Π:

$$\Pi : B \times V_\pi \times E \times \mathcal{P}(A_r) \tag{4}$$

where
B: the set of starting state descriptions, $B \subset S$
V_π: Set of labels for protocols.
E: the set of ending state descriptions, $E \subset S$
$B \bigcap E = \emptyset$

A protocol cannot have disconnected actions in its definition:

$$\forall a \in A_R \; \exists s_s \in \mathcal{S} \wedge A'_R \subset A_R : s_i = s_{i-1}(c_i) \; for \; 0 < i \leq k \tag{5}$$

where:
$a = \langle r_x, \langle s, v, c \rangle, r_y \rangle$
$A'_R = \{a_0, a_1, \ldots, a_k\}$
$a \notin A'_R$ and $a_i \in A'_r$
$a_i = \langle r_{xi}, \langle s_i, v_i, c_i \rangle, r_{yi} \rangle \; for \; 1 \leq i \leq k$
$a_0 = \langle r_{x0}, \langle s_s, v_0, c_0 \rangle, r_{y0} \rangle$
$s_i(c)$: A function that calculates the state description result of applying the operations in c to the state s_i.

A protocol is a directed graph where state descriptions are the nodes which are linked together by actions. Actions represent a system transition from one state description to another in case the action is performed. All starting states in this graph (where there are only outgoing actions) will be part of B and all ending states will be part of E.

Protocol patterns Interaction protocols will present frequent patterns in their structure. These patterns can be, atomic protocol, a protocol sequence, split and merge:

atomic protocol : The most basic kind of protocol, it is composed of a single action. Pre- and post-conditions are the same of the action.

protocol sequence : A protocol composed of one single starting state description and one single end state description.

protocol split : A protocol composed of a set of actions which all share the same starting states.

protocol merge : A protocol composed of a set of actions which after being performed will lead to the same state description.

3.4 Composition of protocols

Using the presented model of protocols a mechanism for composition and modularization of protocols can be defined.
To produce a composed protocol, two protocols can be composed by connecting one or more ending states of the first protocol to the equal amount of starting states of the second protocol. To connect an ending state e to a starting state b, the ending state e that will be the enabler of the starting state b needs to be a subset of it: $e \subseteq b$. This way all constraints demanded by the starting state b will be fulfilled by the ending state e.

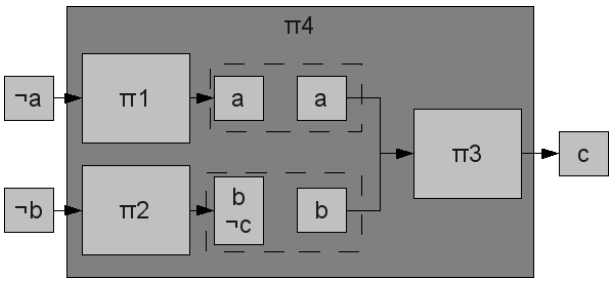

Fig. 2. Protocol composition example, $\pi 4$ is composed of $\pi 1$, $\pi 2$ and $\pi 3$. Dashed lines represent where protocols are linked together.

In Figure 2 an example of composition is shown, in which a protocol $\pi 4$ is composed of three other protocols: $\pi 1$, $\pi 2$ and $\pi 3$. $\pi 1$ is connected to $\pi 3$ because its end state matches exactly a starting state of $\pi 3$: both of them have a valid. $\pi 2$ is connected also to $\pi 3$, because its end state is a subset of the other starting state of $\pi 3$: $\pi 3$ requires b to be valid and the ending state of $\pi 2$ fulfills that constraint, it is the set of states where b is valid and c is not. This is allowed by the state the protocols is connecting to.

4 Implementing Modular Protocols in PIM4Agents

Based on the definitions in Section 3, we proceed to incorporate the Modular Protocol concepts into the Interaction Aspect of the PIM4Agents (see Figure 3). Therefore, we define the *ModularProtocol* class as a specialization of *Interaction*, inheriting then the associations to *ACLMessages*, which represent the previously defined actions, and *Actors*, which represent the roles that take part in the interactions.
The ModularProtocol is a composition of *StateDescriptions*, which are used to represent the intermediate states between ACLMessages and

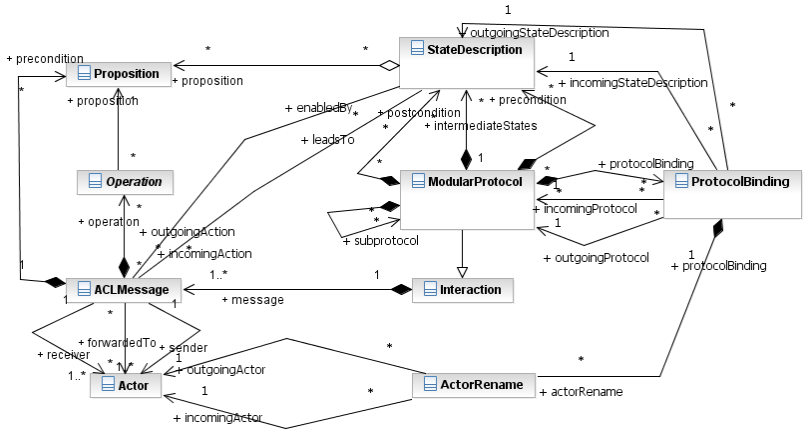

Fig. 3. Partial View of the Modular Protocol Metamodel

also the preconditions and postconditions of the protocols. The StateDescription contains the *Propositions* that are known in the set of states represented by it. When the preconditions of an ACLMessage match a given StateDescription, the ACLMessage is added to the *outgoingActions* of the StateDescription. When the preconditions plus the applied operations of the ACLMessage produce a given StateDescription, this ACLMessage becomes one of the StateDescription *incomingActions*.

The ModularProtocol can be recursively composed of other protocols. This composition is enabled by the StateDescriptions along with the *ProtocolBinding*. In the ProtocolBinding, an incoming protocol is bound to an outgoing protocol, when one of the incoming protocol's postcondition state descriptions contains a subset of the prepositions contained in one of the outgoing protocol's precondition state descriptions (see 3.4). In order to link the roles from the incoming protocol to the corresponding roles in the outgoing protocol, the ProtocolBinding contains a collection of *ActorRename* instances that map each incoming actor with the corresponding outgoing actor.

As mentioned previously, the Propositions, along with the *Operations*, describe the semantics of the ACLMessages and compose the StateDescriptions. The Proposition class represents the base for all propositions and, as depicted in Figure 4 and in correspondance to Section 3.2, it is further specialized by four classes: *Arithmetic*, *TimeoutProp*[1], *Conditional* and *Commitment*. The Operation class is abstract and represents the base

[1] TimeoutProp refers to the Timeout concept presented in Section 3.2 and it is named so to avoid a name conflict with a TimeOut concept used previously in the PIM4Agents

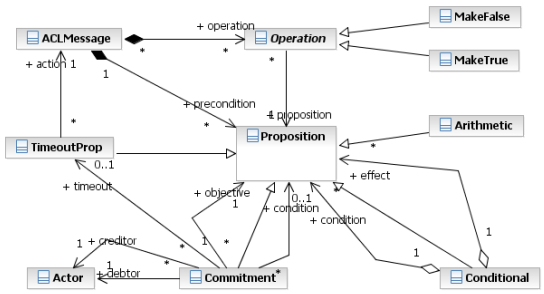

Fig. 4. View of the Propositions hierarchy

for the currently supported operations: *MakeTrue* (+) and *MakeFalse* (−).
In the following section, we will present a small example by using the concrete syntax that corresponds to the concepts introduced in this section.

5 Example

In this section, we revisit the example shown in Section 3.4. In Figure 5, we find the graphical representation of ModularProtocol *Pi4*. It is constituted by 3 subprotocols: *Pi1*, *Pi2*, and *Pi3*, depicted as rectangles. The small squares attached to the sides of each subprotocol represent each of the StateDescriptions that are pre- or post conditions of the given protocol. The black arrows are the graphical representation of the ProtocolBinding objects, linking the postconditions of one protocol with the preconditions of the next.

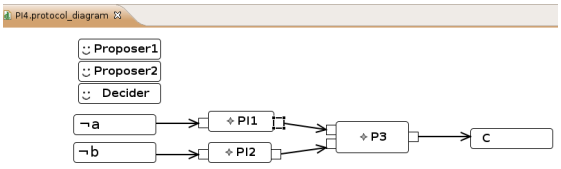

Fig. 5. Graphical View for Protocol *Pi4*

The graphical editor allows the user to also access the internal content of the nested protocol. By double clicking on the nested protocol Pi1,

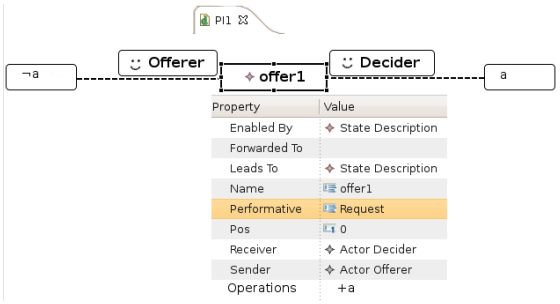

Fig. 6. Graphical View for Protocol *Pi1*

the user is presented with the contents of Figure 6. Pi1 represents the simplest protocol: a protocol with only one action. Therefore, the state descriptions, that represent the states before and after *Offer1* is executed, are also the pre- and postconditions of Pi1.

6 Conclusion

The present work shows how a model for modular protocols is introduced in the metamodel for multi-agent systems PIM4Agents. The utilized interaction protocol model has the advantage of providing modularity and explicit formal definitions for actions and protocols that explain what these concepts are and hence, how they can be combined. A survey of current approaches to interaction protocol models is provided by [7], which shows many approaches to modeling protocols. This idea of having protocols that can be combined to produce new ones has always been the intention for FIPA specifications. It extends simple dialogue games [8] models by focusing not only in the actions that can be performed but also in their consequences. The contribution consists in establishing a clear connection between actions and protocols and using this semantics define clearly how they can be recombined. In [9] a very similar approach is proposed, the main difference is that it proposes a methodology to model protocols that takes the evolution of such systems into account. Our approach instead, uses a model driven approach as used in PIM4Agents.

Given the advantages of this model, it is introduced in a comprehensive metamodel for MAS, providing an agile model for protocols. This model is then given a concrete meaning in the scenario of MAS connecting it to the rest of concepts necessary for MAS. The advantage of this is the possibility of creating new modeling tools that take advantage of the properties of the modular protocol model. An initial implementation has shown that the information present in the model helps to provide a more proactive modelling tool. For instance, the developer can see, by

clicking an ending state of a protocol, which other starting states of other protocols are compatible.
Some other aspects have also to be introduced, like defining transactions or the relation of messages in this model and events. This approach models interactions from a global perspective only, further work will be to see how these protocols can be *projected* to specific roles, in other words, how a specification of a participant can be extracted out of the global model. PIM4Agents has mechanisms to represent also concepts necessary to define this projection, therefore it is an important step to introduce the abstract model of protocols in a concrete modelling tool.

References

1. Bauer, B., Müller, J.P., Odell, J.: Agent UML: A Formalism for Specifying Multiagent Software Systems. In: ICSE 2000 Workschop on Agent-Oriented Software. (2000)
2. Hahn, C.: A domain specific modeling language for multiagent systems. In Padgham, L., Parkes, C.P., Mueller, J., Parsons, S., eds.: Proceedings of 7th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2008). (2008) 233–240
3. Hahn, C., Fischer, K.: The static semantics of the domain specific modeling language for multiagent systems. In: Proceedings of the 9th International Workshop on Agent-Oriented Software Engineering (AOSE 2008). Workshop at AAMAS'08, 13.5.2008. (2008)
4. Warwas, S., Hahn, C.: The contrete syntax of the platform independent modeling language for multiagent systems. In: Proceedings of the Agent-based Technologies and applications for enterprise interOPerability (ATOP 2008)at AAMAS 2008. (2008)
5. McCarthy, J., Hayes, P.J.: Some philosophical problems from the standpoint of artificial intelligence. Readings in nonmonotonic reasoning (1987) 26–45
6. Mallya, A.U., Singh, M.P.: A Semantic Approach for Designing Commitment Protocols. In Eijk, R.V., ed.: Developments in Agent Communication. Volume 3396 of LNAI., Springer (2005) 37–51
7. Miller, T., Mcginnis, J.: Amongst first-class protocols. In: Engineering Societies in the Agents World VIII: 8th International Workshop, ESAW 2007, Athens, Greece, October 22-24, 2007, Revised Selected Papers, Springer-Verlag (2008) 208–223
8. McBurney, P., Parsons, S.: Games that agents play: A formal framework for dialogues between autonomous agents. J. of Logic, Lang. and Inf. **11** (2002) 315–334
9. Desai, N., Choppra, A.K., Singh, M.P.: Amoeba: A Methodology for Requirements Modeling and Evolution of Cross-Organizational Business Processes. ACM Transactions on Software Engineering and Methodology (to appear)

System interoperability analysis by mixing system modelling and MAS: an approach

A.S.Rebai, V.Chapurlat

LGI2P- Laboratoire de Génie Informatique et d'Ingénierie de Production
Site EERIE de L'Ecole des Mines d'Alès, Parc Scientifique Georges Besse
30035 Nîmes cedex 1 – France – tel. (+33) (0)466 387 066
{ahmed-sami.rebai, vincent.chapurlat }@ema.fr

Abstract: In the current economical context, enterprises should work together, collaborate, mix their different skills and competences and exchange their different data all along collaborative processes to respond rapidly and with efficiency to a business opportunity. In this way, due to the required interactions and the different nature of each involved parts (persons, machines, etc…), enterprises have to imagine and to anticipate in which way these interactions will be made. This requires to detect and to limit the loss of performance, integrity and stability the required interactions can induce. That's why it is important to study in an anticipatory way to characterize and to detect causes and effects of a lack of enterprise interoperability within the collaborative process. This paper presents an approach to model a collaborative process in an enterprises network and to simulate its behaviour in order to study its interoperability.

Keywords: Interoperability, Multi Agent Systems, Modelling System, Simulation

Introduction

An enterprise network can be considered as a system of systems i.e. as a set of subsystems which need to interact in a timely manner in order to fulfil a given mission, respecting objectives but remaining independent from other subsystems. [1] defines the systems of systems as a collaboration in which each system wouldn't be adequate to fulfil alone the mission, but together they are capable to do it within an assembly way. Each one of these systems, specified, conceived, implemented and maintained potentially independently of the others, corresponds to a specific project which needs the intervention of distinct actors.
So, these systems will have to interact, and in order to do it in the best manner, they must have a certain degree of interoperability which is defined as the capacity for two (or more) systems to exchange information and to use the information which they exchanged [2]. Here enterprises networks are composed of enterprises or enterprises

parts being involved in this (set of) collaborative process(es) for example in order to apply a common production strategy or to manage health care organisation. All along these processes, enterprises (parts) interact with another. So some requirements must be respected enabling each enterprise part to exchange with these systems without losses or defect. Indeed, each interaction can be at the origin of derived problems (behavioural, functional or structural), losses of performance, and unpredictable emergent behaviour. It becomes then crucial to be able to analyze the interoperability in an anticipatory way in order to limit, at the time of the concretization of the interaction in the real life, the losses, risks and even the damage able to be undergone by the systems which are concerned.

This paper presents an approach to facilitate the detection of interoperability problems, their causes and effects before any concretization. In the current stage of development, the presented work focus on the organisational interoperability which is related to the structure and the organisation of enterprises. This concerns for example the definition of the responsibility and authority which induce and impact the quality of the working conditions. So, the human nature and the organisational behaviours are the main factors which can induce interoperability lack. In other words, interoperability treated here is the potentiality of an enterprise to adapt its organisation (human, equipment, behaviours…) to be able to collaborate with any other enterprise in a better way with minimum efforts and costs. The proposed approach is based on a enterprises network system modelling and analysis framework taking into account principles of system and requirements modelling, techniques for model verification and transformation from system model to multi agent system (MAS) [3] [4] [5] and simulation using MAS.

After illustrating the problematic, the second part of the paper presents briefly the modelling and verification part of the framework. The third part details the current work and focus on the transformation of the system model to a MAS architecture on which simulation allows then to analyze the interoperability.

Organisational interoperability problem illustration and related requirements

Let us take for example the case of an enterprise network in the automobile field which must collaborate together to conceive then to produce blocks of engine whose components need the various fields of competences of these enterprises. So, they must interoperate to ensure the success of this collaboration, not only on the level of the design and the production by taking into account all the technical constraints to which they must face together, but also on the level of the services associated as maintenance, managing the end of the product life cycle.

Interoperability is then a stake which is advisable to analyze before any concretization of an enterprise network. This induces several requirements:

1 - It is convenient to work starting from models. A model is a crucial tool of representation and abstraction of the complexity of the enterprise: comprehension, communication between actors and analysis become facilitated. A model is built for a targeted project. In this case, the model may describe the collaborative process and

the different parts of enterprises involved in this collaborative process i.e. which interact. Indeed, the objective is to model their interactions, their abilities, mission, and behaviours. The concepts issued from system engineering [8], enterprise modelling [9] or more largely systems of systems [1] are then analysed and used. In enterprise modelling domain, in [10], the authors distinguish frameworks architectures as CIMOSA [11] or PERA [12], operational methods as SADT, MERISE [13], GRAI [14], OLYMPIOS [15]. Architectures provide essentially multi view concept allowing representing an organisation by describing from a coherent manner different points of view: functional, informational, decisional and so on. Operational methods provides then numerous concepts and relations allowing to manipulate models in one or several of these views. In the same way, system engineering insists on the relation between systemic domain and system description by proposing also multi view and multi paradigm approaches. Particularly, modelling a system of systems can be facilitated by using frameworks such as DoDAF or Zachman framework [1].

2 – Collaborative process modelling is not sufficient. It is necessary also to describe the interoperability requirements. The current languages and frameworks of modelling are limited to represent this requirement. Some works exists about the development of interoperability rules as proposed in [16] who proposes conceptual enrichments of modelling languages.

3 – Analyse mechanisms of the model may be then available in order to assume if the models check the different interoperability requirements. The goal is not to analyze the behaviour or the structure of each partner taken separately but to analyze the impact of these behaviours and these organizations on their potential interfaces/interactions with the other partners. Indeed, each identified interaction is a potential source of problem (loss of performance, non synchronisation, etc.).

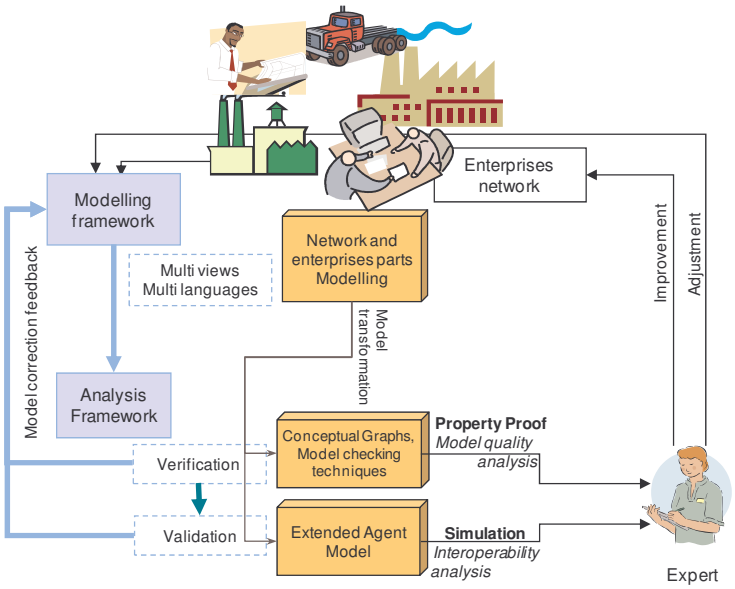

Fig. 1: Synthesis of the approach

The proposed approach schematised in Fig.1 considers two necessary and complementary steps:

- A static manner allowing assuming the quality of the model i.e. a model without misinterpretation or errors due to the modelling process. The property proof as proposed in [17], not detailed here, is then used.
- A dynamic manner: the paper focuses on this type of approach by means of MAS enriched architecture for the analysis of interoperability.

To summarize, the proposed approach aims to provide a modelling framework and an analysis framework presented in the next parts.

Modelling framework

Regarding our problem, the proposed modelling framework has to take into account the followings main requirements:

- To model the networks related partners i.e. to describe the collaboration process concretizing the network objectives and each enterprise part concerned or involved in this process.
- To model interoperability requirements (legal, policies, constraints …) having to be checked by each partner (enterprise, resource, etc.) in order to be able to assume its mission all along the collaborative process.

A system modelling framework called SAGACE [18] has been extended [19] for supporting and guiding the collaborative process model building activity. As proposed in enterprise modelling domain [20], it is a multi views, multi paradigms and multi modelling languages framework summarized in the following. Each view permits gathering and formalizing a given type of knowledge:

1. *Functional view.* The goal is to describe the mission of any partner in the collaborative network but also its own mission, its objectives in terms of performance and the different functions which are concerned by the collaboration.
2. *Structural view.* The goal is to describe the processes and activities of the partner related to the collaborative process, its resources and their organization in order to support these processes.
3. *Behavioural view*. The goal is to describe some potential operational scenarios and configurations of each partners when they will really involved in the collaborative process. This allows fixing some ideas about functioning modes, evolution rules taking into account context and events.
4. *Property view*. This view allows describing the interoperability requirements under the form of properties. A property is a constrained causal relation linking a set of conditions called cause and a set of waited conclusions called effect. Specifying the relation nature and constraints, the cause and the effect requires handling the knowledge described into the three other views. It allows then to enrich the collaborative process model with complementary knowledge linking the partial models issued from each view. Other properties can be used for describing functional or non-functional requirements, such as coherence rules between views and between partial models, semantic rules, attribute evolution laws, expected behaviour, constraints, and objectives.

Each view requires specific modelling languages which are unified, following the MDA principles, both a common and unique collaborative process modelling meta model. This one contains a coherent and sufficient set of concepts and of relations between concepts required for representing the collaborative process. It allows us to adapt and unify some existing and pre-selected modelling languages from the domains of enterprise modelling and systems engineering that were suitable to each view. For example, the functional view uses the objective modelling language proposed by KAOS [21] and the IDEF-0 functional modelling language [22]. The Unified Enterprise Modelling Language [23] allows one to describe the organizational view. Finally, enhanced Functional Flow Block Diagrams (eFFBD) [24] permit to describe operational scenarios in the behavioural view.

Analysis framework

As presented before, the analysis framework aims:

- From a static point of view i.e. independently from any temporal constraints, to check the coherence of the different models and to assume its quality regarding the study objectives.
- From a dynamic point of view i.e. taking into account the possible evolution (or operational scenarios) to test if these scenarios are credible and if potential emerging scenarios can be feared.

Step 1: static analysis

This step consists first in checking the coherence of the model (coherence between different models coming from different views, internal coherence of the model regarding its meta model and coherence of the usage of concepts and relations in the model). Second it consists to prove that some interoperability requirements are respected. For example, it is necessary to check if capability and capacity of a human resource are adequate taking into account the activity in which this resource will be involved.

This is done by proving in a formal and automated manner that a set of specified properties are verified by the model. If this is not the case, the analysis process must provide a counterexample that indicates the reasons for which the property is unsatisfied. The modeller may then detect modelling errors, mistakes, or misunderstandings, thereby increasing the level of confidence on the model.

To do this, [19] proposes formal rewriting mechanisms to translate the system model towards a formal model. Verification tools such as model checkers or theorem provers [25] can be then used. However, the proposed checking technique is based on a formal knowledge representation and analysis language called Conceptual Graphs [26].

Step 2: Dynamic analysis

The goal of this step is to simulate the behaviour of the model of the enterprises network related as possible to its reality in order to assess the validity of the model. Obviously, the simulation using MAS is seen as the best way to do it rather than formal methods for many reasons. First, the increasing number of interactions between different actors of the enterprise network when considered at the level of individual resources, including elements of organisational knowledge, working capital, human resources, machines, work in progress, computers and buildings, behave the system much more complex with unpredictable events. Second, the MAS can model the human resource, its behaviour, skills and so on by using the BDI (Belief, Desire, intention) paradigms and then particular MAS architecture [27].

MAS for analyzing interoperability

MAS model

Different types of agents are:

- Required for the classical functioning of the simulation platform. JADE [28] has been chosen for many reasons. It proposes a set of pre defined architecture of agents and communication protocols which can be adapted and enriched for our purpose. In particular, communication channel agent provides the road for the interactions between agents and memorizes the messages exchanged. The management system agent demands the other agents to register to be seen by all other agent and it manages their life cycle. The Directory facilitator agent memorizes descriptions of the agents as well as the services which they offer. The agents can write their services in this agent and ask it to discover the services offered by other agents.
- Required for representing the different enterprises entities (processes, resources ...). Different agent models are then proposed. They are enriched by the BDI profile and by linking them to other agents respecting particular constraints. The following meta model presents a partial view of the used conceptual agent model.

These proposed agents are synthesised in Fig. 2:

- **System agent**: is a general concept of agent which recovers all types of agents. Each agent can send and receive messages described by the *Messages* throw a communication protocol defined by the *Communication protocol*. One or more behaviour characterize each agent and it consists of three parts:
  - A set of current *beliefs*, representing information the agent has in connection with its current environment and his own state;
  - A set of *desires*, representing the options available to the agent, i.e. the actions it can do;
  - A set of current *intentions*, representing the goals towards which the agent is engaged and towards which it ask other agents to react.

  BDI profile is written as a set of rules defining how each agent will evolve when receiving messages from other agent. This BDI profile is defined by translating

the behavioural model described in the behavioural view of the pointed out system model. As said before, this behavioural model is described as a finite state machine (FSM) or by using behavioural modelling language such as eFFBD. So taking into account the behavioural model of the modelled entity, the way the agent responds to the messages can be then one shot, cyclic or according to a finite state machine. Last, as proposed in the property view of the modelling framework, each entity is characterized by a set of properties [29]. A property describes an expectation which have to be checked taking into account its context and its evolution stage: it allows detecting bad behaviours, errors or mistakes. A property is used here for describing interoperability requirement as presented before. For example, an organisation unit named 'Maintenance Team' may be always composed of two technicians or a process may stop before end of the day. So System Agent describing an entity is characterized by the same set of properties and at each stage of its evolution, it must verify these properties. If a property cannot be checked by the Interoperability analysis agent, it can be interpreted as a lack of interoperability and a possible source of problem. The checking mechanisms are proposed by [29]. All the other agents are built by using the same architecture.

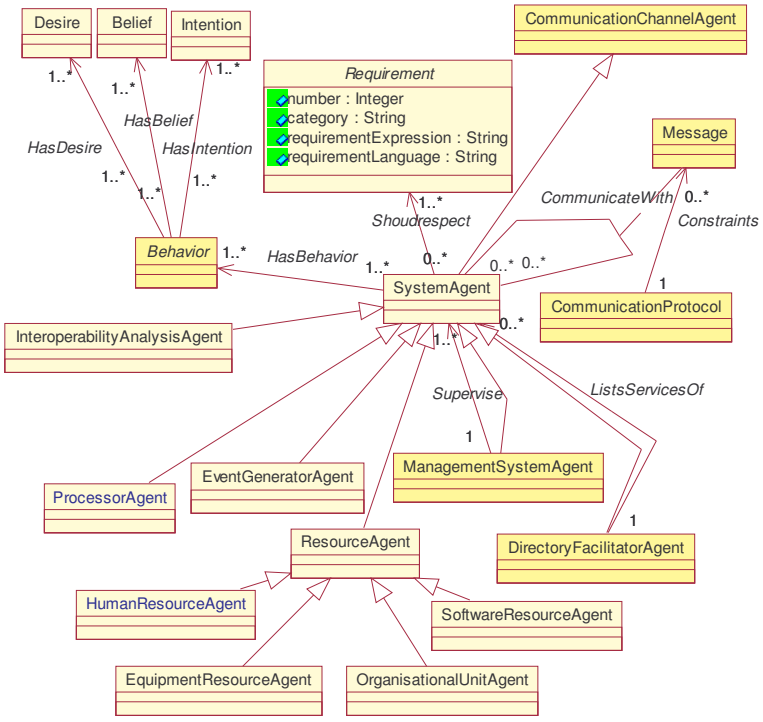

Fig. 2: Agent meta model partial view

– **Resource agent**: represents the different resources composing the enterprise network structure.

- **Processor agent**: represents different processor, activities… of the enterprise network. So processor agent behaviour is directly described in the behavioural views attached to each system components (activities, processes, resources ...).
- **Event generator agent**: generates events described in the system model and having to occur all along the system life cycle taking into account hazardous generation algorithm.
- **Interoperability analysis agent**: observe all the messages exchanged between all the other agents, and check at each evolution step if the interoperability requirements are violated.

From system model to MAS

Starting from the model of the system, the translation step shortly illustrated in Fig.3 allows us to translate this model into MAS following a MDA technique as proposed by [30] [31].
The left part of this figure in grey colour represents concepts coming from the system model. The right part in yellow colour represents their corresponding concepts in the MAS model. All the translation rules cannot be presented due to the limited place. For example, the 'system' concept has as corresponding concept 'AgentSystem'. The concept 'Belief' found its correspondence into 'environment' and 'ability' concepts. This is because the set of the beliefs of each agent is composed by two type of information: the first one is about its environment and the second one is about itself and its capacity which is presented in the system model by 'ability'. So, in each evolution step during the simulation, each agent receives information and performs actions and updates its beliefs i.e. its information about the environment and itself. The 'intention' of each agent will be deduced from 'mission', 'finality' and 'objective'. It represents what the agent must do. During the simulation, these sets of intentions changes according to the evolution of the agent beliefs. The concept 'Desire' has no corresponding one because it is deduced from the intention and beliefs. In short, a BDI agent must update its beliefs with information which comes to it from its environment, to decide which options are offered to it and this will consists on its desires, to filter these options in order to determine new intentions and to choose the action it will do according to all this information. The manner the agent will react can be one shot, cyclic or FSM depending in the message received. In the case of FSM behaviour, information is taking from its different configurations described in the system model.

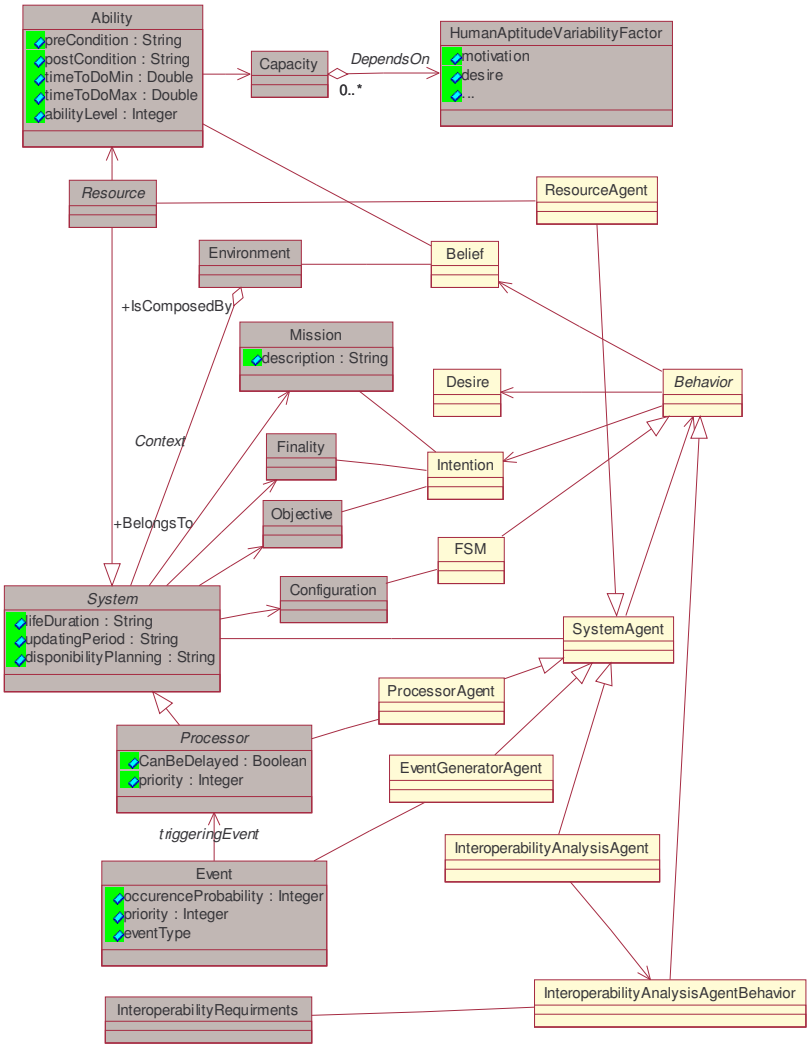

Fig. 3: Translation from system model to MAS

MAS in action

The objective of the MAS execution i.e. the simulation, is first to validate the obtained MAS. Indeed, some scenarios are already described in the system model. Following the used system modelling framework SAGACE, a scenario describes a potential deployment of resources, in given configuration and context of the environment, during processor execution taking into account the limited capability

and capacity of resources, the possible parallelism or synchronisation between processors during their execution, etc. So, MAS simulation allows user to simulate if these existing scenarios can be obtained by simulation. If it is the case, the MAS may be then used for reaching a second objective.
This consists now to test if new scenarios can be obtained and particularly scenario during which some interoperability problems may occur and cause prejudices to collaborative process behaviour. In this case, the interactions i.e. the messages exchanged between agents, are autonomously and dynamically established, dropped and re-established depending on the collaboration needs at any time by taking into account self-organisation principles. However, during their execution each agent must respect the set of properties which constrain its behaviour (efficiency), its structure (integrity) and its functionalities (for example its ability and capacity in the case of human resource). If a property cannot be verified in the current state of the agent taking into account the agent context, the obtained scenario can then describe a possible source of interoperability problem.
The simulation process is then the following. As proposed before, simulating the collaboration process requires assigning a Resource Agent to each concerned and potential resource of the process. This one is described in a Processor Agent. Last, a unique Event Generator Agent is built. It gathers the list of possible events coming from the environment (external events such as customer enquiries, order, energy break, etc.) or coming or required by each behavioural view of each system entity (internal event such as start of a machine, etc.).
The Event Generator Agent computes a possible schedule of these events. It sends messages according to event nature and event probability occurrence to the other agents in order to activate them. The other agent will then run and exchange messages. The interoperability analysis agent will observe and analyse these messages by checking interoperability requirements at each evolution moment.
Each execution is of course unique. The autonomy left with the agents makes it possible to obtain traces of different executions and then different statistics. Indeed, it will be possible to analyze the similarity of the original process that should be executed with the obtained scenarios according to the behaviour of the agents and especially human resource agents whose behaviour is impacting by their belief, desire and intention which corresponds on the real word to their motivation, skills, etc. The information exchanged throw some messages like incomplete information, incoherent information, message arrived too late to be useful, etc will enables us to analyse the causes of this facts and the way they impact the interoperability.

Conclusion and Future Works

This paper presents an approach focusing on modelling and analysis of the interoperability in enterprises networks. It insists particularly on dynamic analysis mechanisms based on simulation of MAS model. In the current step of project development, the enriched agent model, rewriting principles and simulation process have to be implemented. For this, different behavioural models of each agent, the

different messages exchanged and how will this impact the enterprise network interoperability have to be studied.

References

1. Luzeaux, D., Ruault, J.R. : Ingenierie des systèmes de systèmes: méthode et outils, Hermes Lavoisier, [in French], (2008)
2. IEEE: A compilation of IEEE standard computer glossaries, standard computer dictionary, (1990)
3. Brandolese, A., Brun, A., Portioli-Staudacher, A.: A Multi-Agent approach for the capacity allocation problem, International Journal of Production Economics, Vol. 66, pp. 269--285, (2000)
4. Ferber, J.: Les systèmes multi-agents : Vers une intelligence collective, Editions InterEditions, [in French], (1995)
5. Weiss, G.: Multiagent systems: a modern approach to distributed artificial intelligence, MIT Press, Cambridge, MA, USA, (1999)
6. Chen, D., Dassisti, M., Elvesaester, B.: Enterprise Interoperability –Framework and knowledge corpus – Advanced report, INTEROP Deliverable DI.2, (2006)
7. ISO 14258: ISO 14258 –Industrial Automation Systems – Concepts and Rules for Enterprise Models, ISO TC184/SC5/WG1, (1999)
8. INCOSE: System Engineering (SE) Handbook Working Group, System Engineering Handbook, A «How To» Guide For All Engineers, http://www.incose.org/, (2004)
9. Vernadat, F. : Techniques de Modélisation en Entreprise: Applications aux Processus Opérationnels, Paris, Edition Economica, [in French], (1999)
10. Pourcel, C., Gourc, D. : Modelisation d'entreprise par les processus : Activité, organisation & applications, Cepadues, [in French], (2005)
11. AMICE: CIMOSA: Open System Architecture for CIM, Berlin, Springer, (1993)
12. Williams, T.: The Purdue Enterprise Reference Architecture, Computers in Industry, vol. 24 n° 2-3 pp. 141—158, (1994)
13. Tardieu, H., and al : La Méthode Merise : Principes et outils Editions d'Organisation, [in French], (2000)
14. Doumeingts, G., and al : Production management and enterprise modelling, Computers in Industry, vol. 42 n° 2-3 pp. 245—263, (2000)
15. Braesch, C., and al : La modélisation systémique en entreprise, Hermès, [in French], (1995)
16. Blanc, S., Ducq, Y., Vallespir, B. : Enterprise Interoperability : Interoperability Characterization Using Enterprise Modeling and Graph Representation, Springer London, (2007)
17. Chapurlat, V., Aloui S.: How to detect risks with a formal approach? From property specification to risk emergence, in proceedings of Modelling, Simulation, Verification and Validation of Enterprise Information Systems (MSVVEIS), Paphos, Cyprus, (2006)
18. Penalva, J. M.: SAGACE method: the modelling of human designed systems, COMETT'93, (1993)
19. Aloui, Saber., and al : System engineering and enterprise modelling for risks management: application to the drug circuit in a university hospital. URMPM - Union of Risk Management for Preventive Medicine 2nd American Congress, Montreal, Canada 14-15 June, (2007)
20. Vernadat, F. B.: Enterprise Modelling and Integration: Principles and Applications, Chapmann & Hall, (1996)

21. Bertrand, P., Darimont, R., Delor, E., Massonet, P.,Van Lamsweerde, A.: GRAIL/KAOS: an environment for goal driven requirements engineering, 20th International Conference on Software Engineering, IEEE-ACM, Kyoto, April (1998)
22. Menzel, C.P., Mayer, R. J.: The IDEF Family of Languages in Handbook on architectures of information systems, Bernus P., Mertins K. and Schmidt G. ed., Berlin, Springer, (1998)
23. UEML: Deliverable D3.1: Requirements analysis: initial core constructs and architecture, Unified Enterprise Modelling Language UEML Thematic Network - IST-2001-34229, www.ueml.org, (2003)
24. Oliver, D. W., Kelliher, T. P., Keegan, J. G.: Engineering complex systems with Models and Objects, McGraw-Hill, (2004)
25. Yahoda: Formal verification tools overview web site, http://anna.fi.muni.cz/yahoda/, (2003)
26. Sowa, J.F.: Conceptual structures: information processing in mind and machine, New York (U.S.A.), Addison-Wesley, (1984)
27. Wooldridge, M., Jennings, N.: Intelligent Agents: Theory and Practice, Knowledge Engineering Review, (1995)
28. JADE : JADE platform: Java Agent Development Framework, http://jade.tilab.com/, (2000)
29. Chapurlat, V., Kamsu-Foguem, B., Prunet F.: Enterprise model verification and validation: an approach, Annual Review in Control, Volume 27, Issue 2, pp 185--197, (2003)
30. Bézivin, J., Gerbé, O.: Towards a Precise Definition of the OMG/MDA Framework, ASE'01, (2001)
31. MDA, Model Driven Architecture (MDA), Architecture Board ORMSC, Joaquin Miller and Jishnu Mukerji Eds., (2001)

An Interoperability Framework for the Negotiation-Based Coordination of Adaptive Supply Web Agents

Christian Russ[1], Alexander Walz[2]

[1] Dacos Software GmbH, Science Park 2, D-66123 Saarbrücken, Germany
christian.russ@dacos.com

[2] University of Stuttgart, Graduate School of Excellence advanced Manufacturing Engineering, Breitscheidstr. 2c, D-70174 Stuttgart, Germany
alexander.walz@GSaME.eu

Abstract. This paper suggests a design of an interoperability framework supporting the identification of potential interaction partners as well as the mapping of heterogeneous ontologies between them in a B2B-enabled supply network domain. This framework can be used for realizing the negotiation-based coordination of self-interested agents by using a described negotiation protocol for bilateral price negotiations together with a corresponding adaptive negotiation module. If autonomous supply agents are provided with the described negotiation module they are able to adapt their negotiation strategies to dynamically changing negotiation partners and market conditions automatically. The effects of the adaptation processes of the intelligent business agents are simulated and evaluated on the basis of a prototypical agent-based supply network for computer manufacturing. A first focus is centered on the emergence of niche strategies within a group of cooperating supply agents and a second on the effects of different parameterizations of the learning process on the overall profit of the supply network.

Keywords: Distributed Artificial Intelligence, Multiagent Systems, Simulation Modeling and Output Analysis, Coherence and Coordination, Intelligent Agents, Evolutionary Learning, Experimental Economics, Agent-based Supply Chain Management, Supply Networks, Coordination Mechanism Design, Bilateral Negotiation, Genetic Algorithms.

1 Introduction

A *supply chain* is a chain of possibly autonomous business entities that collectively procure, manufacture and distribute certain products. Since today's markets are highly dynamic, current supply chains are forced to respond to consumer demands more accurately, flexibly and quickly than ever before.

In order to stay competitive, supply chain partner companies are forced to form supply chains on the basis of more flexible and co-operative partnerships. For these reasons, so-called supply webs (see Laseter [4] and Porter [5] - i.e. B2B-enabled dynamic networks of supply chain units - will replace today's static supply chains to an increasing extent.

Agents offer the advantage of being able to automatically form and manage such changing partnerships since they can autonomously perform tasks on behalf of their users and in doing so reduce transaction cost. But in the case of B2B-enabled dynamic

supply networks, there may occur a “clash of ontologies” between heterogeneously designed agents that come together in such a dynamic and open environment in order to negotiate and exchange resources not necessarily use the same ontologies.

Hence, an *interoperability infrastructure* has to be established consisting of *interoperability services* (as directory, mediator and ontology services) and efficient *coordination mechanisms* (e.g. specialized negotiation or auction mechanisms) that help the heterogeneous supply chain agents to co-ordinate their local activities.

Therefore, for ensuring their operability we have suggested in this paper firstly an interoperability framework provided by a *interoperability service agent (ISA)* encapsulating directory, mediator and ontology services to the supply chain agents, secondly, an *automated protocol for bilateral price negotiations* and thirdly a corresponding *negotiation module* that enables evolutionary adaption of negotiation strategies to changing negotiation partners in the described competitive context. The negotiation module contains elaborated learning capabilities, enabling a self-interested agent to learn from his negotiation history in order to adapt his negotiation strategy. These three elements together provide a suitable interoperability framework for interactions in an open supply web domain.

2 Interoperability Framework

Independent and heterogeneously designed supply chain agents may generally use different internal denotations and representations for goods and services. For this reason directory, mediator and ontology services have to be provided to them in order in order to ensure their interoperability.

2.1 Interoperability Service Agent

Thus we propose a global *interoperability service agent (ISA)* encapsulating directory, mediator and ontology services and providing them to the supply chain agents.

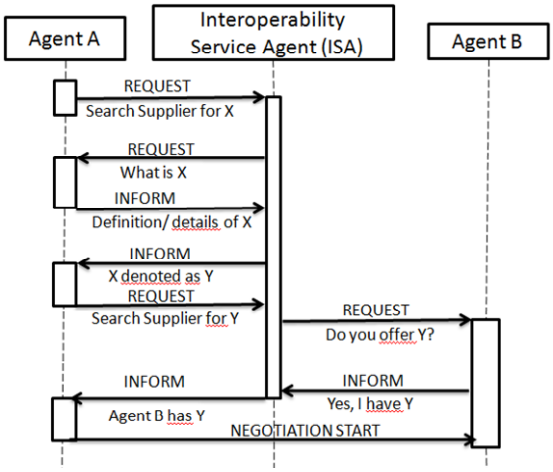

Fig 1. Matchmaking and ontology mapping by the interoperability service agent

The ISA allows the agents to find other agents with intersecting goals (i.e. who offer or buy the same goods) by a directory service. This service administrates all the agents within a simulation scenario with their unique names, addresses, agent type and properties, i.e. capabilities, offered and demanded goods etc. On the other hand it acts as a negotiation mediator by providing also a bilateral price negotiation protocol with a common ontology usable by all agents as described in the next paragraph.

Additionally, the ISA may administrate several possible ontologies for the input and output goods in the supply web domain and thus is able to map differing ontologies of heterogeneously designed agents, which is important for the matchmaking between agents with intersecting goals.

To find out if a mutually beneficial transaction can be carried out, each agent may select another agent in the network to start a bilateral negotiation by using the ISA-services a shown in Fig. 1.

If, e.g. the ISA has to pass on a request for buying a good X from an agent A to a supplier agent B and semantics behind the denotation X is not known for the ISA, the ISA may ask the agent A about the detailed description of good X. Since it may be the case that this good is known in the reference ontology used by the ISA but is denoted by another expression (e.g. "Y") because the agents possess local heterogeneous ontologies. Therefore A has to send all his knowledge about the specification of X to the ISA. Then an *ontology mapping service* of the ISA tries to find a match between the things agent A knows and the expressions it uses for and the knowledge and expressions of the reference ontology used by the ISA. For realizing such an ontology mapping service [8] suggests a mixture of two different methods, namely N-Grams and WordNet. N-Grams computes a lexical relation between two words whereas WordNet calculates a semantic similarity. The combination of the two concepts is necessary because e.g. the words automobile and car are lexically very different from each other whereas the meaning of both is closely the same and therefore WordNet finds a strong relation.

2.2 Negotiation Protocol

Since the internal price ideas of an autonomous agent are not visible to another agent in advance, the agents have to gather information about the negotiation space, while they are negotiating with each other. The negotiation protocol used is a kind of strategic choice protocol that is also adopted by humans in economic real-life negotiations and is described in detail in the following.

Negotiation Acts

In the bilateral negotiation process, all agents are provided with the same action alternatives derived from Pruitt's *strategic choice model* [6]. This model states that humans show specific behavior in economic negotiation scenarios and select in every negotiation round from among five basic strategies, namely

1. ***unilateral concession***: decreasing the distance between conflicting positions,
2. ***competitive behavior***: remaining with one's position and trying to argue the other party out of his position by pressure, arguments, threats, etc.,

3. ***coordinative behavior***: Bilaterally collaborating and dissolving the controversy,
4. ***idleness***: not continuing the negotiation and making no counteroffer, or
5. ***demolition***: withdrawing unilaterally from the negotiation.

Eymann [2] states that these basic building blocks of human negotiation strategy can be further reduced to the following three *negotiation action alternatives* sufficient for negotiations in the examined multi-agent domain:

1. ***Accept***: the price proposal of the other agent is accepted and the transaction is conducted. The buyer pays the end price to the seller and receives the product.
2. ***Propose***: the agent at turn does not agree to the price proposal of his opponent and makes a new proposal on his part. This new proposal can be equal to his last proposal (no concession in this round of negotiation) or be newly calculated.
3. ***Reject***: an agent breaks off the negotiation. Both agents thus have to search for other negotiation partners to fulfill their needs.

Strategy Parameters

For modeling complex and not easily predictable strategic behavior on the basis of the described action alternatives in automated negotiations we use six *strategy parameters* that determine the negotiation strategy of an agent. These can take on values from the interval [0;1] and are stored in a so-called ***genotype***, a data structure suitable for processing by a genetic algorithm:

1. *acquisitiveness (A)*
2. *delta_change (DC)*
3. *delta_jump (DJ)*
4. *satisfaction (S)*
5. *weight_memory (WM)*
6. *reputation (R)*

An agent possesses several genotypes that are evolutionarily adapted to varying negotiation partners and market conditions as described in section 4. Each parameter of a genotype influences non-deterministically an agent's individual negotiation behavior. Since the negotiation strategy of an agent is implemented as a finite state machine where these parameters define the probabilities of changing from one of the negotiation states (accept, propose, reject) to another and additionally define how quickly, how often and to what extent concessions are made etc.

The *acquisitiveness* of an agent defines the probability that he will offer a unilateral concession on his next "move", i.e. as the seller lowering his asking price. This parameter taking on a value of 1 would prohibit an agent from making any price concession while a value of 0 would motivate him to always concede.

The *delta_change* parameter defines the step size of a monetary concession by specifying a percent value by which the price distance between one's last price proposal and the counter offer of an opponent is reduced. An agent calculates his *individual* step size for a concession at the beginning of a negotiation by using

$$current.stepSize_A = (asked\_price - offered\_price) * del\_change\ .$$

This keeps the negotiation mechanism *symmetric* since neither sellers nor buyers are handled in a preferred manner (for attributes of a coordination mechanism see [3]).

The *delta_jump* parameter defines the margin an agent wants to realize. The higher the value of *delta_jump*, the higher is the aimed margin between the buying costs for input goods and the demanded selling price will be. For this purpose, *delta_jump* modifies the first price proposal of agent A in a price negotiation for a good as follows:

$$proposal = agreement * (1.0 + del\_jump)$$

where agreement equals the price of the last deal for the good.

The fourth parameter, *satisfaction*, defines the probability that an agent aborts the negotiation and thus ensures that a negotiation does not continue on and on. The abort probability after the n^{th} negotiation round amounts to $(1 - p\_satisfaction)^n$.

To avoid individually nonsensical behavior the agents have a learning function to detect unreasonable price proposals during a negotiation. Therefore an agent stores transaction prices, i.e. the end prices of successful negotiations, from his negotiation history in a data structure *memory* and calculates an internally *"sensed" market price (SMP)* for each good of interest. This is necessary because there is no central institution for declaring market prices. The information stored in *memory* is used to compute his *smp* with exponential smoothing. Thereby, the parameter ***weight_memory*** specifes how fast market changes have influence on the market price.

```
// update our memory of initial prices
memory = offeredPrice * weight_memory + memory * (1-weight_memory);
```

On this basis, at negotiation start each agent checks the first price proposal of his opponent against his SMP. All counter-proposals lying between the SMP and its doubled value are estimated as uncertain and a possible negotiation abort is tested according to *p_satisfaction:*

```
if (offeredPrice >= memory) {
// ...then random reject check
If (randomNumberIsHigherThan(p_satisfaction)) {
        reject = true;
}
// reject all offers more than double memory
if (memory != 0 && offeredPrice > 2 * memory) {
        reject = true;
}
```

All counter-proposals exceeding the doubled SMP are directly rejected to avoid extortion offers. The last parameter *reputation*, defines the probability of finishing a deal correctly according to the reputation of a negotiation partner in the system.

The values of these six parameters describe completely the negotiation behavior and are adapted by the following process in the course of a sequence of negotiations.

2.3 Negotiation Module

Each negotiation module of an agent possesses a *genetic pool* of genotypes. This pool

contains numerous genotypes that are employed in negotiations. After a negotiation has been finished a *fitness value* is calculated for the genotype in depending on the negotiation outcome. Then the genotype is stored in combination with the ascertained fitness value as *plumage* in a data structure called *population*. The sizes of pool and population can be flexibly set for the negotiation module of each agent.

After the start of a bilateral negotiation, the first step of an agent is to choose a genotype - determining his strategy for this negotiation - out of his pool of genotypes. Then, both agents negotiate until one of them aborts the negotiations or their price proposals cross and they make a mutually beneficial deal. After a successful negotiation both agents calculate a fitness value for the used genotype and store the resulting combination of genotype and estimated fitness as a so-called plumage into the population data structure.

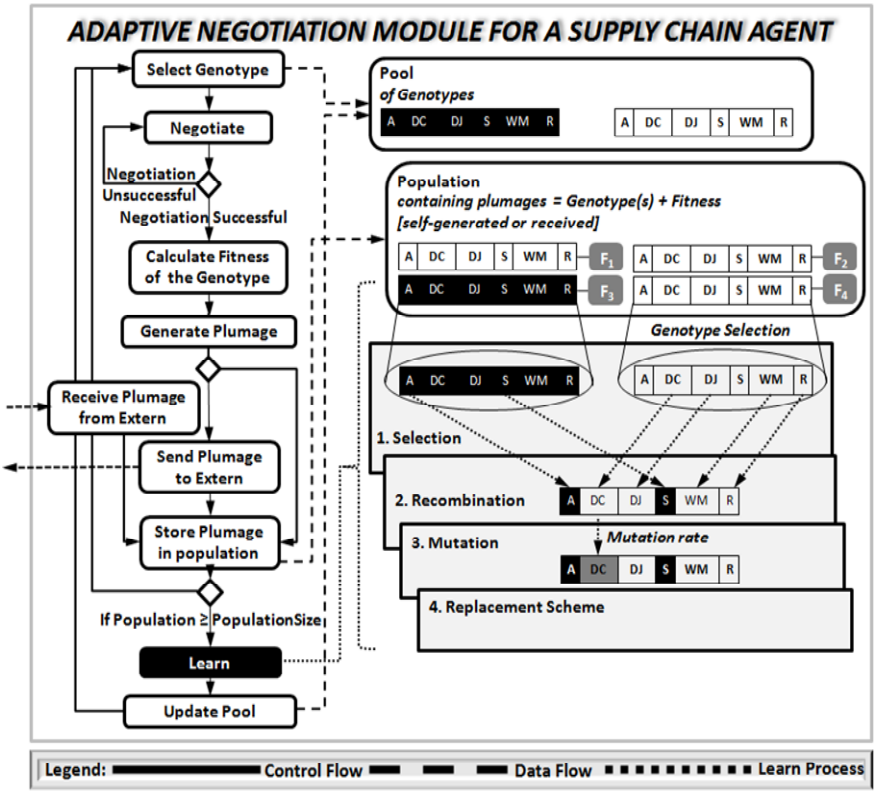

Fig 2. Co-action of negotiation, information exchange, and learn process

If their information exchange mode is set to external or mixed, they will afterwards send the resulting plumage to other agents, receive plumages from other allied agents and store the self-calculated as well as the received plumages in their population. If the number of stored plumages is larger than the population size the agents will start their learning process by using their individually parameterized evolutionary learning mechanism. This process is sketched in the Fig. 2.

3 Parameterization of the Negotiation Module

When the learning process is started all plumages within the population are assigned to a *selection method*, which selects the plumages with the best fitness values and assigns as many plumages to a *recombination process* as the pool size allows. In this recombination process selected genotypes are recombined to new genotypes. Optionally, the newly built genotypes can be modified by probabilistic *mutation* after the recombination step. In the last step of the learning process, the old population of the agent is deleted and the newly generated pool is assigned to the agent according to his specific replacement scheme. After that the agent may start new negotiations. The possible settings for the learning process are described in the following.

3.1 Parameterization of the Information Exchange

An agent learns either only by himself (*internal learning mode*), i.e. does not use the experiences (in the form of plumages) of other agents. Or an agent has "colleagues" that exchange information about successful finished negotiations with him in such a way that he can use this information in his next evolution step (*external learning mode*). An agent may also use both modes in parallel (*mixed learning mode*).

3.2 Parameterization of the Learn Process

Fitness Calculation

After a successfully finished negotiation, i.e. a mutually closed deal, the fitness value for the genotype used in the negotiation is calculated. Our negotiation module offers the agents' designer the following *fitness calculation methods:*

1. The *price_minus_average (PMA)* fitness function uses the margin between the current average price of a good and the current price:

$$fitness = avgPrice\text{-}currentPrice$$

2. The *percental_average_proceeds (PAP)* method takes the duration of a negotiation into account by dividing the PMA fitness value by the average price times the number of rounds in which this genotype was used.

$$fitness = \frac{average\mathrm{Pr}ice - current\mathrm{Pr}ice}{average\mathrm{Pr}ice * roundsforCurrentGenotype}$$

3. The *percental_absolute_proceeds (PAB)* method divides the value of the PMA variant by a fixed basicPrice times roundsForCurrentGenotype.

$$fitness = \frac{basic\ \mathrm{Pr}ice - current\ \mathrm{Pr}ice}{basic\ \mathrm{Pr}ice * roundsforC\ urrentGeno\ type}$$

4. The *percental_mean_proceeds (PMP)* method uses the mean value (mediumPrice) of the starting price proposals of both partners in the negotiation.

$$fitness = \frac{medium\ \mathrm{Pr}ice - current\ \mathrm{Pr}ice}{medium\ \mathrm{Pr}ice * roundsForC\ urrentGeno\ type}$$

Selection

1. *Binary competition (BC)*: Two randomly selected individuals are compared and the one with the higher fitness value is selected and copied in the new population. This is done repeatedly until the new population has reached its defined maximum size.
2. *Roulette-Wheel-Selection (RWS)*: Each individual I is assigned a section on a wheel based on his fitness value according to the formula:

   $\alpha = 360 \frac{f(I_k)}{\sum_{n=1}^{N} f(I_n)}$, with

   α : Angel assigned to the k[th] individual

   $f(I_K)$: Fitness value of the k[th] individual

   N : Number of individuals

3. *Deterministic Selection (DS)*: Based on the fitness value of the individual i an expectation value is calculated:

   $$Expectation\ E(X) = f_i \frac{N}{\sum_{k=1}^{N} f_k}$$

4. *Deterministic Average Selection (DtS)*: In the case that the same genotype has been used in several negotiations and thus is contained in the population more than once this method calculates the average fitness value for each individual genotype before the selection starts. The individual with the best fitness value comes directly into the new population; the worst one is deleted and the remaining genotypes are selected in the same manner as in the DS method.
5. *Deterministic Average Selection with deletion of the worst individual (DAS):* The new population is filled up in the same way as with the standard deterministic average selection until the number of individuals in the new population equals pool size minus the value of BestIndividualsToSurvive. For the last place a new genotype is generated as the mean value of each gene of the genotypes already included in the new population.

Recombination

This crossover of individuals with good fitness values is a kind of macro mutation which creates jumps in the search space and therefore helps to find a better solution.

1. *n- Point- Crossover (nPC)*: The two best individuals are taken out and with a probability, the crossover probability, these two are recombined or they are put back in the population unchanged. If the two individuals are recombined they are cut at ‘n’ randomly chosen positions and linked crossover. The new generated genotypes replace their parents in the population.
2. *Random Crossover (RaC)*: Two individuals are selected and for each gene it is decided which is taken for the new one according to a probability identified by preferBetterparent.

Mutation

Its main target is to keep the diversity in the population. Usual modifications according to the Gaussian distribution are suggested in [1]. In our negotiation module all values are in the interval [0;1] and calculated according to the formula

$$Gen = Gen + gaussWidth * nextGaussian(),$$

where nextGaussian returns a random number and gaussWidth is the breadth of the Gaussian distribution. For each genotype in the population it is decided if it is mutated - and if so it is changed at exactly one position.

Replacement scheme

After the creation of the new pool it has to be decided what to do with the old one. By simply deleting the old pool there is a risk that all new genotypes are worse than their parents. To protect the best individuals two replacement schemes are used:

1. *Elitism:*To protect the best 'n' individuals from being modified the individuals with the highest fitness value are transferred into the new population unchanged.
2. *Weak Elitism:* As before, the best 'n' individuals are transferred into the new population but before this they can be mutated, as described above.

4. Simulation Framework and Evaluation Results

4.1 The MACSIMA Multiagent Simulation Framework

To examine the dynamics within a supply network built up by autonomous agents using the proposed interoperability framework as well as the described negotiation module, we have used MACSIMA (Multi Agent Supply Chain SIMulAtion Framework). The MACSIMA framework has been implemented in Java and offers a set of generic supply chain agent types for instantiating supply network scenarios:

1. *resource or supplier agents (R_i)* supply raw materials to the network.
2. *producer agents (P_i)* stand in the middle of the value chain and buy *raw materials* or *semi-finished goods* from resource agents or other producer agents as input goods to their *production function* and offer their output goods for purchase.
3. *consumer agents (C_i)* stand at the end of the added value network and buy products from the producers. They cannot run out of money, but however, have a *consumption function* that specifies their maximal willingness to pay.

MACSIMA offers a GUI that simplifies the definition of topologies and enables one to parameterize the learning process to much more detail as compared to the limited learning features of precedent approaches (e.g. [2] and [7]).

4.2 Impacts of Evolutionary Adaptation

In our simulation runs conducted so far we have mainly concentrated on different instantiations of a five-tier-supply-network for computer manufacturing. A first scenario has been defined whereby genetic adaption was initially turned off for all

agents. All agents of the scenario were provided with the same static strategy parameters (*acquisitiveness = 0.5*, *delta_change = 0.25*, *delta_jump = 0.15*, *satisfaction= 0.75*, *weight_memory = 0.2* and *reputation = 1*) – except for the processor producers whose acquisitiveness parameter was set to 0.51 instead of 0.5. The price fluctuations in the resulting simulation run are presented in Fig. 3 (a).

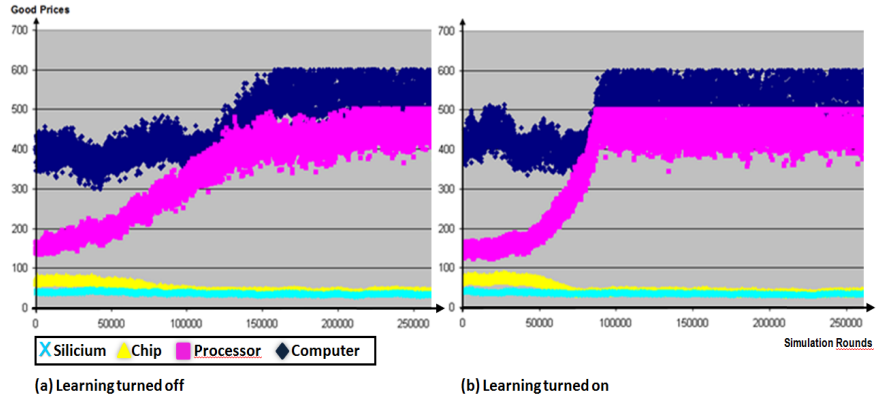

Fig. 3. Price fluctuations in (a) a setting with static strategy parameters and genetic adaptation turned off for all and (b) turned on for processor producer agents

It can be seen that already the slight modification of the acquisitiveness parameter of +0.01 leads to a crucial strategic advantage for the processor agents and enables them to force their negotiation partners on adjacent tiers to significant concessions.

In a second simulation scenario we keep the settings of the foregoing scenario but exclusively provide the 10 processor producer agents with the ability to learn from previous negotiation outcomes. Their new learning capability empowers them to benefit now much faster from their competitive advantage as shown in Fig. 3 (b).

4.3 Emergence of Niche Strategies

As we can see in the Fig. 4, this effect is not achieved because all of the 10 autonomously negotiating processor agents follow the same strategy.

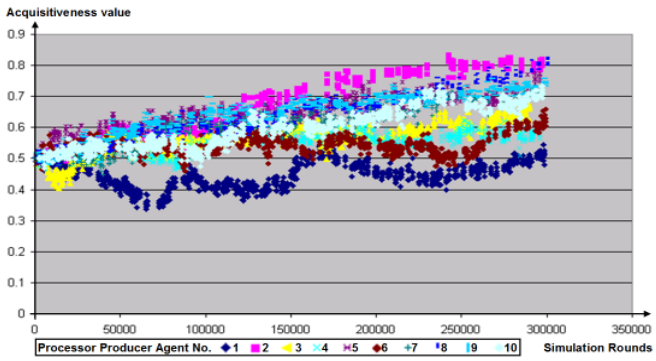

Fig. 4. Emergence of niche strategies

Instead of this, some agents of the processor tier benefit from following a niche strategy consisting in keeping the value of one's own acquisitiveness parameter slightly lower than the majority of the agents of the same type - or at least by stabilizing it at the lower spectrum of their parameter value range.
Agents of the same type but of lower acquisitiveness than their colleagues can benefit from this fact and increase their sales since their opponents expect no concession from a processor agent and thus accept quickly, if unexpected concessions are made.

4.4 Parameter Settings for Maximizing Overall Profit and Turnover

We have examined the question of whether there exists a parameterization by which - if applied by all the agents in a network - social welfare maximizing effects may be expected. Therefore we have defined 50 simulation scenarios including two "baseline" scenarios each with a different parameterization of the learning process that is applied by all agents. In the first baseline setting learning was turned off for all agents and in the second the STDEA mechanism [7] was used by all agents. A summary of the evaluation results is given in Fig. 5. It can be seen that an infelicitous parameter choice results in a waste of welfare in such a way that overall turnover and sales may fall under the level of both baseline settings. Otherwise, an expert parameterization outperforms the first baseline by approximately 400 percent.

The top parameterization we have found so far has the following settings: *<pool_size=3, population_size=40, information_exchange=mixed, selection_method=roulette:wheel, recombination=n-point-crossover, mutation_rate=0.5, Gaussian_width=0.01, replacement_scheme=elitism>*.

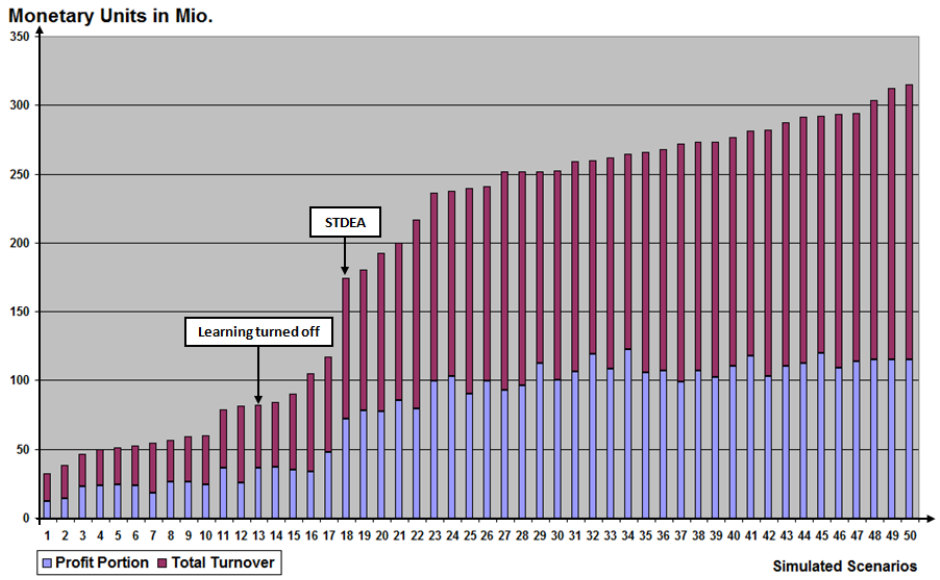

Fig. 5. Expert parameterization of the learning mechanism maximizes profit and turnover

5. Conclusion

We have described the design of an interoperability framework supporting the identification of potential interaction partners as well as the mapping of their possibly heterogeneous ontologies. This framework is suitable for realizing the negotiation-based coordination of self-interested agents in a B2B-enabled supply network domain. For this purpose we have also presented a negotiation protocol for bilateral price negotiations together with an adaptive negotiation module for taking part in corresponding negotiations. We have described the design and implementation of a negotiation module that can be used by autonomous agents for adapting their negotiation strategies. The effects of the adaptation processes of the intelligent agents have been simulated and evaluated using the MACSIMA framework for realizing a prototypical agent-based supply network for computer manufacturing.

We have outlined some evaluation results with a first focus on the emergence of niche strategies within a group of cooperating agents at one tier of a supply chain for computer manufacturing. After that we have centered a second focus on the effects of the application of different learning mechanism parameterizations on the overall profit of supply networks. Our simulation results show that depending on the parameter setting for the learning mechanism the overall profit and turnover of the supply network can vary about 1000 percent.

References

1. Back, T., Fogel, D. B., and Michalewicz, Z., 1997. The Handbook of Evolutionary Computation. Oxford University Press, 1997.
2. Eymann, T., 2000. AVALANCHE - Ein agenten-basierter dezentraler Koordinationsmechanismus für elektronische Märkte, Inaugural-Dissertation, Albert-Ludwigs-Universität Freiburg im Breisgau, 2000.
3. Fischer, K., Russ, C., and Vierke, G. 1998 Decision Theory and Coordination in Multi-agent Systems. Research Report RR-98-02, DFKI, 1998.
4. Laseter, T. M. 1998. Balanced Sourcing: Cooperation and Competition in Supplier Relationships. Jossey-Bass, ISBN: 0787944432, October 1998.
5. Porter, A. M. 2000. Supply management in 2010: Experts see big future for e-procurement. In: Purchasing online, March 23, 2000. http://www.manufacturing.net/magazine/purchasing/
6. Pruitt, D. G. 1981. Negotiation Behavior, Academic Press, New York, 1981.
7. Smith, R.E., Taylor, N., 1998. A Framework for Evolutionary Computation in Agent-Based Systems, In: Looney, C., Castaing, J.: Proceedings of the 1998 International Conference on Intelligent Systems, S. 221-224. 1998.
8. Teixera, D.D., Cardoso, H.L., and Oliveira, E. 2008. An Ontology-Mapping Service for Agent-Based Automated Negotiation. In K. Fischer, A. J. Berre, J. P. Müller & J. Odell (eds.), *Proceedings of The AAMAS'08 Workshop on Agent-based Technologies and Applications for Enterprise Interoperability (ATOP)*, pp. 1-12, Estoril, Portugal, May 13, 2008.

Specification of strategies for negotiating agents

René Schumann, Zijad Kurtanovic, and Ingo J. Timm

Information Systems and Simulation,
Goethe University Frankfurt am Main
Robert-Mayer-Str. 10, 60325 Frankfurt am Main, Germany
[reschu|zijad|timm]@informatik.uni-frankfurt.de

Abstract. Negotiations are an important part of today's inter-enterprise business processes. Interoperability in negotiations has addressed more technical aspects, so far. Common protocols and shared ontologies can provide sufficient solutions. Actually the interoperability on the process level is more challenging, that is how the multiple attributes of a negotiation issue can be assigned simultaneously respecting all constraints between the attributes in a way that the outcome is acceptable for both human negotiators.
In this paper we present a negotiation model that allows agents to negotiate on behalves of human negotiators. Therefore we formalize negotiation strategies and use a meta model based on ECORE as a framework for specifying negotiation strategies. To allow efficient negotiations among agents, we present a technique for efficient representation and reasoning about these negotiation strategies.

Key words: automated negotiation, tradeoff strategies, MDA

1 Introduction

Negotiations are an important part of today's inter-enterprise business processes. According to Pruitt [1] a negotiation is a process, by which two or more parties try to reach a mutually acceptable agreement on some matter. So far these negotiations are hard to automate. Nevertheless, the field of automated negotiations is an important field in the multiagent system (MAS) research [2, 3].
Thus designing systems that implement business process including negotiations between two or more companies is a hard task. This interoperability problem is neither on the technical level. Common communication protocols and shared ontologies can be applied and thus lead to an interoperability on the system level. Interoperability becomes critical on the process level. Within a negotiation a set of parameters, like price or quality, has to be determined simultaneously. This is complicated by the fact that commonly an agent does not know how the different parameters depend on each other for its opponent and what parameter configurations are acceptable for him.

Within a negotiation encounter there may exist more than one deal satisfying both parties [4, 5]. Thus when the negotiation runs in a conflict because of the

parties' distinct interests and preferences, interaction can only proceed by making new proposals with the aim of coming to a mutually acceptable agreement [5]. Those new proposals can be found via tradeoffs, which are an important aspect of negotiations in the human behavior [6, 4] and have been adopted for software agents [7, 5, 8] as well. A tradeoff between two negotiation attributes is a combination of attributes values. The main idea of a tradeoff thereby is to improve one attribute while worsening of other attributes in return [8].

In this paper we encourage the notion of agency that agents act on behalf of their owners, thus the agent should follow the negotiation strategies specified by its owner, the human negotiator. Thus the negotiation may have not an optimal outcome from a game theory point of view, but the agents behave in concern with its owner. The problem of *acquiring* such knowledge from the human negotiator is often left open [3, 8]. But this information is an essential requirement making use of automated negotiation in practical settings. Within a negotiation strategy it is specified how the agent can generate alternative offers based on tradeoffs, and how the agent behaves within a negotiation. This allows automated negotiations within inter-enterprise business processes. Here we present a specification language that allows the human negotiator to specify the negotiation strategy in a comprehensive way, without prerequisite that he has knowledge of the techniques used to implement its agent. Because typically the human negotiator is not a programmer, capable to develop it's agent by himself. The specification language is based on the ECORE meta model which is part of the Eclipse Modeling Framework (EMF) [9]. This allows an easier usage of model driven architecture (MDA) concepts to automatically transform the specified negotiation strategy in software code that can be used by the software agent. Thus the owner of the agent can be ensured, that the agent will act, as he has specified, without a potential erroneous and expensive process of transforming the specification into the agents code by hand.

The rest of this article is structured as follows. In the next section we present the underlying negotiation process. We explain the entire protocol and outline, what role the negotiation strategies have. In section 3 we refine the concept of negotiation strategies by giving a formal definition and show how those strategies can be defined using ECORE . We show that a negotiating agent using tradeoffs has to solve a prioritized fuzzy constraint satisfaction problem (PFCSP) to generate suitable offers or counter offers. As the PFCSP is in general too complex to be solved during a negotiation process that proceed under real time conditions we demonstrate in section 4 how the negotiation process can be implemented efficiently. The entire automated negotiation process is demonstrated in section 5. Finally we draw our conclusion and outline future work.

2 Negotiation model

In this work we use a negotiation model presented by Lou et al. [5]. It is a simple negotiation setting with a bilateral negotiation. Two roles are defined: a buyer and a seller agent. Both agents negotiate a contract with a number of attributes,

like price, quality, delivery or payment date. Each of the agents has a global preference function that orders all permutations of all possible outcomes of the negotiations. The agents operate in a semi-competitive environment. This is reflected by their behavior strategies which are based on the *principled negotiation approach* [6]. That is, that they try to weaken their position only minimally e.g. by minimal information disclosure, minimal relaxing their desires [5].

In the following we will outline the negotiation protocol and the negotiation strategies relevant aspects.

2.1 Negotiation protocol and agent's behaviors

The negotiation protocol is based on the alternating offers protocol [10]. Seller agent's behavior protocol is presented in figure 1. The states represent the al-

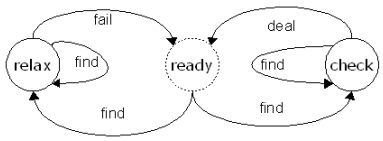

Fig. 1. Seller agent's behavior protocol

lowed performatives he can send in a given negotiation context, except the dashed one - this is interpreted as the initial state, in which the seller agent is ready to take proposals from buyer agents. The edges represent the performatives of the buyer agent that can be received during a negotiation encounter. During the negotiation the buyer can specify a set of constraints that each potential offer has to met.
When the performative *find* is received the negotiation is initiated and the agent can answer with performatives *check* or *relax*. He uses the performative *check* when he finds an offer satisfying the currently published constraints of the buyer. If there exist no such offer, the seller ask to *relax* at least one of the constraints, so that he can find a suitable offer.

The buyer agent's behavior protocol is presented in figure 2. The states and edges are defined analogous to figure 1. The buyer agent is the initiator of a negotiation and he does so by sending a *find*-performative to the seller agent. While doing so he publishes the constraint with the highest priority to the seller[1]. After he sends a *find*-performative he expects a *check* or *relax* performative from the seller agent. If a *check*-performative is received, he checks the proposed offer and answers with *deal* if it is acceptable, i.e. all constraints are met and the acceptance threshold is exceeded. Otherwise he criticizes the offer by publishing the constraint that is violated. If multiple constraints are violated the one with the highest priority is chosen.

[1] If there are different constraints with equal priority one is randomly chosen.

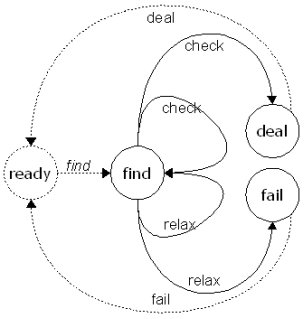

Fig. 2. Buyer agent's behavior protocol

2.2 Negotiation strategy

The decision made in the agents behavior specification, like to decide which counteroffer should be send, or if an offer should be accepted base on data given by the user. These data is encoded in the negotiation strategy. Thus the complexity of the negotiation strategies is independent from the complexity o f the negotiation protocol or the agents behavior.

Of course not every deal is acceptable for the agents. Each agent has a minimal acceptance threshold that must be exceeded. This guarantees that e.g. the seller agent does not agree to a deal with a negative profit. The seller agent has a list of products. For each product a set of value vectors of negotiable attributes is associated. The set of resulting possible deals can be computed by:

- a tradeoff strategy, given by the user,
- the global preference function, and
- user's threshold that specifies what preference value is needed at least.

A tradeoff strategy specifies what combination of attributes form an acceptable deal for the user. It specifies a set of tradeoffs and preferences over attribute values and attributes value combinations. A tradeoff is a relation between two attributes of the negotiations. It defines that in favor for worsening one attribute the other has to improve at a certain rate.

The buyer agent has a requirement model for each product he is interested in, which expresses his specifications about the desired products. This is done in form of fuzzy prioritized constrains. Fuzzy constraints reflect a natural way of modeling user's requirements [5]. Often different priorities can be found in real world settings, thus these fuzzy constraints are prioritized [11]. These requirements are derived from a tradeoff strategy given by the human negotiator, the global preference function and user's thresholds, as well.

So far it becomes clear that the specification of tradeoff strategies is an essential part automating negotiations as outlined in this article. Therefore we detail tradeoff strategies in the next section.

3 Tradeoff strategies

Within a tradeoff strategy all information about tradeoffs for a negotiation are encoded. As sketched previously the tradeoff strategy is a set of pairs[2] of attributes which are in a tradeoff relation and a set of *independent* attributes. A tradeoff function defines how much one attribute can be worsened, in favor for improving the other. According to [8] this can be formalized as follows:

Definition 1
Let the value set of the attribute x be defined with $X = [l_x, r_x]$, and let the value set of y be $Y = [l_y, r_y]$. Then the function is called tradeoff function between X and Y if it is continuous, monotonic *and met the* boundary condition. *The boundary condition assures, that if one attribute is assigned to the best value, the other attribute has to be made worse [8].*
The pair (x, y) is called a tradeoff pair.

For each tradeoff pair a preference function is defined, which specifies the preference over the tradeoff alternatives. Tradeoff alternatives are value combinations of the relevant tradeoff pair [8]. Independent attributes are not in a tradeoff relation with other attributes. To each of them a preference function is associated. A tradeoff strategy represents a *directed forest*, this is formalized in definition 2.

Definition 2
Let the negotiation attributes as nodes and the tradeoff pairs as directed edges be given. The direction of an edge is defined by the tradeoff function (see definition 1). If this function has the form $\tau : X \rightarrow Y$, there exists an edge from node X to Y. Then a tradeoff strategy *represents a* directed forest*:*

- *Let a be a negotiation attribute. All values of a's value set A are in a preference ordering relation $\preceq \subseteq A \times A$: the preference direction is* descending *if smaller values are allowed,* ascending *otherwise.*
- *To each tradeoff pair (a, b) the following is associated, with A and B as the value sets of a and b respectively:*
  - *A tradeoff function $\tau : A \rightarrow B$ in terms of definition 1.*
  - *A tradeoff preference function $p : A \times B \rightarrow [0, 1]$, which assigns to each tradeoff alternative a preference value. It reflects a trapezoid formula of three segments (analogue to the preference function in [8]) to describe the increasing, steady and decreasing preference over tradeoff alternatives.*
- *Independent negotiation attributes are trees with only one node. For each such negotiation attribute a a preference function is associated $p : A \rightarrow [0, 1]$, with A: $\forall a, b \in A : \ a \preceq b \Leftrightarrow p(a) \leq p(b)$.*

An example of a tradeoff strategy is shown in figure 3. Hereby the value sets are marked in the lower level of the nodes and user's *best* tradeoff alternatives are labeled at the edges. The edge going from the trapezoid to the first node is labeled with the priority of the tree to which it connects. A more complex

[2] Experience shows that tradeoffs between a pair of attributes are the most common [4], hence we focus only on them.

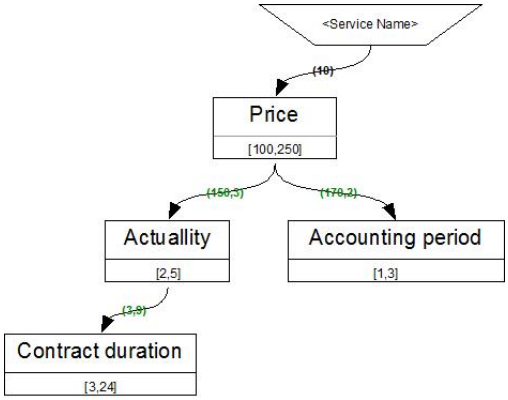

Fig. 3. Exemplified tradeoff strategy

tradeoff strategy can have more complex tradeoff nets, which is sketched in figure 4. In fact it is ensured that all tradeoff strategies can be represented as a forest. This ensures some formal but also informal benefits. A tradeoff strategy can be visualized in a *clear* and *accustomed* way to the users. Due to the acyclic structure there cannot exist inconsistencies, which may be introduced by cycles. This reduces the complexity specifying and validating those strategies. Moreover this forest structure allows the use of efficient algorithms for reasoning about the tradeoff strategies [12].

As already mentioned, a tradeoff strategy can be seen as a set of constraints. In the negotiation scenario it's often the case, that the constraints are *fuzzy* and have different level of importance [13, 5]. In this regard the tradeoff strategy can be modeled as a *prioritised fuzzy CSP* [11] in the following way:

- There is a set of variables with an associated value set; to each negotiable attribute (i.e. variable) a value set is associated.
- From each tree a fuzzy constraint is derived; for negotiation attributes connected as a tree - one fuzzy constraint.

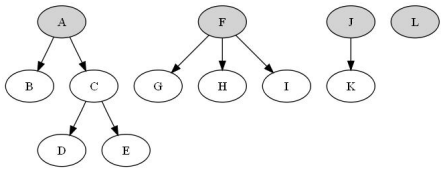

Fig. 4. Tradeoff strategy as directed forest: nodes represent negotiation attributes, edges tradeoff relations. L is an independent attribute

– A priority function $\rho : T \rightarrow [0, +\infty)$ assigns to each tree (i.e. fuzzy constraint) a level of importance.

To gain the tradeoff strategies from the user we have developed a metamodel for negotiation strategies, and thus tradeoff strategies in particular. Therefore we used the meta-metamodel Ecore of the Eclipse Modeling Framework (EMF) [9]. Basing on EMF allows us have the ability to integrate a graphical editor for tradeoff strategies in the near future and currently use the concepts of model driven development for a more rapid development. Using EMF the metamodel can be transformed to code. Prior to this a generator-model is needed - a platform-specific model which supplies the EMF-generator with information like the connections between multiple Ecore-models, the name of the generated files, referenced Ecore-models etc. [14]. With use of the generated editor, a user can model his *negostrategy*-based negotiation strategies visually, validate it against the metamodel and save it as XMI-documents. These documents can be transformed into data-objects representing the negotiation strategy used by an agent specifying its negotiation strategy.

4 Efficient reasoning for negotiations

As outlined in the last section a tradeoff strategy forms a PFCSP. Problems of this form are, as all CSPs, hard to solve [15]. To ensure an efficient negotiation process a suitable representation and reasoning technique has to be identified. The main idea to cope with this complexity is to compute a representation set of acceptable deals prior to the negotiation. This representation set can be encoded within a relational representation, as a set of tables. In fact it is necessary for every tree of the tradeoff strategy (see e.g. figure 4) to generate one table. The remaining information not kept in the tables like e.g. the preference directions of attributes' value sets needed for generation of critics are provided through the code model of the *negostrategy*-metamodel. Thus during the negotiation process reasoning about offers and the generation of critics can be done querying the representation set, e.g. using SQL statements.

The computation of the set of acceptable deals is not done at compile time, because context aspects, like the fact with whom a negotiation is done, can be respected in the negotiation strategy. If these aspects have to be integrated in the relational representation the resulting tables would be significantly larger, as all possible contexts would have been respected.

The relational representation is a set of tables where each table represents a fuzzy-constraint, that is equivalent to a set of tradeoffs with different preferences over some negotiable attributes. By joining the tables according to the tree structure the valid combinations of attributes' values can be derived and ordered according to the preference function of the agent. Each row in a table represents an assignment over all negotiation attributes in the tradeoff tree with respect to the approximated tradeoff functions which is called a *tradeoffconsistent* assignment. For each cell in a column (i.e. negotiation attribute), a subset of the value set is derived on the basis of the preference direction of the negotiation

attribute and the value contained in that cell. Thus to each row i.e. *tradeoff-consistent* assignment, the agent can derive combinations of interval sets, which represent a spectrum of possible tradeoffs over negotiable attributes represented by that table. This allows the agents to search more efficiently for a mutually acceptable solution i.e. over a set of potential solutions, rather than over single point solutions each round. For each tree the procedure is as follows:

- First a representation set is generated from the root negotiation attribute, containing a list of values from its value set.
- along the directed edges i.e. tradeoff relations the representation lists of all remaining negotiation attributes are derived through the approximated tradeoff functions.
- the preference of each row is computed, with the global preference function.

An example is shown in figure 5, this corresponds to the set of acceptable of the negotiation strategy specified shown in figure 3. In this table all possible deals that satisfy all tradeoff conditions are shown ordered by their preference value.

ACCOUNTINGPERIOD	PRICE	ACTUALITY	CONTRACTDURATION	PREF	PRIOPREF
1.86	160.0	2.9	10.5	0.97...	0.97666...
1.64	145.0	3.2	8.4	0.89...	0.89666...
2.06	175.0	2.75	12.75	0.89...	0.89666...
2.25	190.0	2.6	15.0	0.73	0.73
1.43	130.0	3.8	6.6	0.61...	0.61333...
2.44	205.0	2.45	17.25	0.54...	0.54666...

Fig. 5. Table of all possible attribute combinations of the attributes price, actuality and contract duration

5 Automating Negotiations a case study

In this section we demonstrate the specification of a negotiation. We have designed a simple scenario where the seller offers access to an information service, that the buyer wants to subscribe. Attributes of the contract are

- price (PR)
- actuality of the data (AC)
- contract duration (CD)
- accounting period (AP)

From the seller's perspective these attributes can have the following values: The price can be in a range of [120,270] €, of course a higher price is preferred. The delivered data can have an actuality of 1,2,4 or 6 hours. As more accurate data is more expensive, older data is preferred. The seller assumes its optimal

ration between profit and accuracy gaining €170,- for two hour old data. Possible contract durations are 6,12,18 or 24 month, longer durations are preferred. Accounting periods can have a length of 1,3,4 or 6 month, shorter periods are preferred, not given a credit to the customer.

From the buyer's perspective the attributes have other desired values and preferences, of course. The price should be in the interval between [100,200] €, and of course a lower price is preferred. Actuality of the data should be between two and five hours, more accurate data is preferred and a higher price is acceptable. A fair ratio between accuracy and price for the buyer is paying €150,- for three hours old data. The contract duration can be in an interval between [3,24] month, where a shorter duration is preferred being more flexible. For a better (for the buyer a lower) price the buyer might be willing to accept longer contract durations. Acceptable accounting periods can be one to three month. Longer periods are preferred, but for a better price shorter ones can be accepted.

The resulting tradeoff strategies can be specified as described in section 3. In fact the tradeoff strategy of the buyer was shown in figure 3. The tradeoff strategy of the seller is shown here in figure 6. Using the negotiation metamodel we have

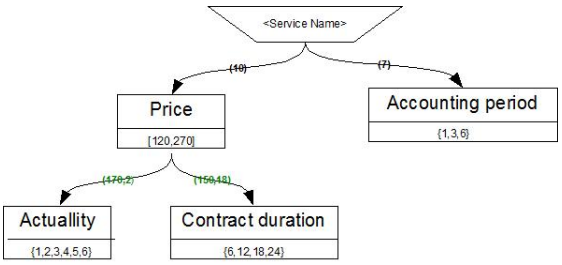

Fig. 6. Seller tradeoff strategy

generated an EMF-editor that allows specifying these negotiation strategies in a comprehensive way. After the negotiation strategies have been specified they can be saved persistently using the XMI format.

Before the agents start a negotiation they have to build a relational representation of the set of acceptable deals induced by the tradeoff strategies. This set defined by the buyer's tradeoff strategy has already been presented in figure 5[3]. The resulting tables for the seller are shown in figure 7. If the two agents try to negotiate about subscription terms for the information service the resulting negotiation trace is shown in table 1. The buyer starts the negotiation by selecting the first top row of the table 5 which contains the most preferred combination of attributes' values according to his tradeoff strategy. According to his behavior

[3] The directed forest representing buyer's tradeoff strategy contains only *one* tree which corresponds to *one* table

PRICE	ACTUALITY	CONTRACTDURATION	PREF	PRIOPREF
165.0	2.4	16.5	0.96...	0.965000...
180.0	1.9	15.0	0.885	0.885
150.0	3.6	18.0	0.855	0.855
195.0	1.75	13.5	0.74	0.74

ACCOUNTINGPERIOD	PREF	PRIOPREF
1.0	1.0	1.0
1.5	0.95	0.985
2.0	0.84	0.952
2.5	0.74	0.922
3.0	0.63	0.889
3.5	0.53	0.859

Fig. 7. Relational representation of seller's tradeoff strategy. His thresholds are already considered, thus only acceptable tradeoffs are shown

strategy he always tries to minimize the revelation of private information, thus revealing only one constraint to the seller i.e. he requests a deal for a price $\leq$ €160,-. The seller uses the SQL-select command and finds an offer that satisfies buyer's price constraint. He sends these appropriate contract conditions to the buyer and asks him to evaluate his offer. In round 2 the buyer evaluates that the offer is not acceptable because of some violated conditions e.g. for the offered price a better actuality of data and shorter contract duration is expected. In consequence he asks the seller to find another offer satisfying the price and the additional violated actuality constraint. As the seller agent has no fitting offer[4] he asks to relax these constraints. In doing so the buyer lowers his expected satisfaction degree with the deal. Finally, after 2 more unacceptable offers from the seller in rounds 4 and 6, a deal is reached in round 8.
If, for example, the buyer had more strict thresholds which limit his tradeoff ability, then the negotiation could fail.

6 Conclusion

In this article we present a negotiation model that allows complex negotiations among two agents acting on behalves of the human negotiators. This capability allows automating negotiations as part of complex inter-enterprise business processes, thus enabling interoperability on the process level. In doing so, it was pointed out that the formal specification and modeling of negotiation strategies is important. In particular tradeoff strategies are of relevance.

The main contribution of our work is that we have defined a formal representation of negotiation strategies, for agents acting on behalve of human negotiators. Thereby we use the ECORE meta model, to allow the negotiators themselves to describe the negotiation strategy. The transformation into executable code that can be used by the agent is done via automatic transformation gaining the advantages of a formal definition and the ideas of MDA. Thus the expensive and possibly erroneous process of encoding these strategies by hand is avoided.

Due to the lack of space we have omitted more details of the meta-modeling language for negotiations that we have specified. This meta-model includes a

[4] In this case the SQL-select statement using buyer's conditions applied on his relational representation shown in figure 7 returns an empty result set.

Round 1	Buyer	Performative: Find Constraint: PR $\leq$ 160
	Seller	Performative: Check (PR:150,AC:4,CD:18,AP:1)
Round 2	Buyer	Performative: Find Constraint: PR $\leq$ 160 $\wedge$ AC $\leq$3
	Seller	Performative: Relax
Round 3	Buyer	Performative: Find Constraint: PR $\leq$ 145 $\wedge$ AC $\leq$3
	Seller	Performative: Relax
Round 4	Buyer	Performative: Find Constraint: PR $\leq$ 175 $\wedge$ AC $\leq$3
	Seller	Performative: Check (PR:165,AC:2,CD:18,AP:1)
Round 5	Buyer	Performative: Find Constraint: PR $\leq$ 175 $\wedge$ AC $\leq$3 $\wedge$ CD $\leq$ 13
	Seller	Performative: Relax
Round 6	Buyer	Performative: Find Constraint: PR $\leq$ 190 $\wedge$ AC $\leq$3 $\wedge$ CD $\leq$ 15
	Seller	Performative: Check (PR:180,AC:2,CD:12,AP:1)
Round 7	Buyer	Performative: Find Constraint: PR $\leq$ 190 $\wedge$ AC $\leq$3 $\wedge$ CD $\leq$ 15 $\wedge$ AP $\geq$ 2
	Seller	Performative: Check (PR:180,AC:2,CD:12,AP:3)
Round 8	Buyer	Performative: Deal

Table 1. Full negotiation trace of buyer and seller (PR: price, AC actuality, CD contract duration, AP accounting period)

graphic specification, too. As we have outlined, the presented specification of tradeoff strategies is very expressive on the one hand, but on the other hand can cause complex computational efforts, as tradeoff strategies can be described as a prioritized fuzzy constraints satisfaction problem. Thus efficiently representations and reasoning is necessary to achieve implementable solutions. We have shown that a relational representation of the set of acceptable deals induced by the negotiations strategy can be computed be used during the negotiation.

An underlying vision of our project is to allow the human negotiator, who is responsible for the negotiations, not for its technical implementation, to specify his negotiation strategy in a form that can be transformed automatically into the reasoning knowledge of the agent. Therefore we will extend our tooling. As the negotiation specification meta model includes a graphical notation, we are going to develop a visual editor for the specification of negotiation strategies. Making it more convenient for the human negotiator. Moreover it is intendet to automate more phases of the specification of multiagent negotiations using MDA princi-

ples. So further steps can be the specification and automated transformation of negotiation protocols.

As the presented method follows an engineering approach to specify negotiation strategies it is an open question what expressional power the presented approach for specifying negotiation strategies has, compared to other approaches like bargaining theory.

References

1. Pruitt, D.G.: Negotiation behavior. Organizational and occupational psychology. Acad. Pr., New York, N.Y. (1981)
2. Wooldridge, M., Ciancarini, P.: Agent-oriented software engineering: the state of the art. In: First international workshop, AOSE 2000 on Agent-oriented software engineering, Secaucus, NJ, USA, Springer-Verlag New York, Inc. (2001) 1–28
3. Kraus, S.: Strategic negotiation in multiagent environments. Intelligent robots and autonomous agents. MIT Press, Cambridge, Mass. (2001)
4. Steele, P.T., Beasor, T.: Business negotiation: A practical workbook. Gower, Aldershot (1999)
5. Luo, X., Jennings, N.R., Shadbolt, N., Leung, H., Lee, J.: A fuzzy constraint based model for bilateral multi-issue negotiations in semi-competitive environments. Artificial Intelligence Journal **148**(1-2) (2003) 53–102
6. Fisher, R., Ury, W.: Getting to yes: Negotiating agreement without giving in. 2. edn. Mifflin, Boston (1991)
7. Mudgal, C., Vassileva, J.: Bilateral negotiation with incomplete and uncertain information: A decision-theoretic approach using a model of the opponent. In: In Klusch and Kerschberg (Eds.) Cooperative Information Agents IV, LNAI, Springer-Verlag (2000) 107–118
8. Luo, X., Jennings, N.R., Shadbolt, N.: Acquiring user tradeoff strategies and preferences for negotiating agents: A default-then-adjust method. Int. J. Hum.-Comput. Stud. **64**(4) (2006) 304–321
9. Foundation, T.E.: Graphical editing framework (gef). http://www.eclipse.org/modeling/emf, accessable at 08.01.2009
10. Osborne, M.J., Rubinstein, A.: Bargaining and markets. Economic theory, econometrics, and mathematical economics. Acad. Press, San Diego, Calif. (1990)
11. Luo, X., Lee, J., Leung, H., Jennings, N.R.: Prioritised fuzzy constraint satisfaction problems: axioms, instantiation and validation. Int Journal of Fuzzy Sets and Systems **136**(2) (2003) 155–188
12. Russell, S.J., Norvig, P.: Artificial intelligence: A modern approach ; [the intelligent agent book]. 2. ed. edn. Prentice Hall series in artificial intelligence. Prentice Hall, Upper Saddle River, NJ (2003)
13. Dubois, D., Fargier, H., Prade, H.: Possibility theory in constraint satisfaction problems: Handling priority, preference and uncertainty. Applied Intelligence **6** (1996) 287–309
14. Budinsky, F.: Eclipse modeling framework: A developer's guide. The eclipse series. Addison-Wesley, Boston, Mass. (2003)
15. Ghallab, M., Nau, D., Traverso, P.: Automated planning : theory and practice. Elsevier, Kaufmann, Amsterdam et al. (2004)

Selection of Resources For Missions Using Semantic-Aware Cooperative Agents *

Murat Sensoy, Wamberto Vasconcelos, Geeth de Mel, and Tim Norman

Department of Computing Science, University of Aberdeen, AB24 3UE, Aberdeen, UK
{m.sensoy, w.w.vasconcelos, g.demel, t.j.norman}@abdn.ac.uk

Abstract. Different organizations carry out various missions. Some missions may be crucial for the organizations, and require critical but scarce resources. Scarcity of these critical resources and importance of the missions create an incentive for the organizations to cooperate by sharing their resources with an expectation of carrying out their missions successfully even if the resources in hand are limited. In this paper, we propose a multiagent framework where organizations' missions are semantically described, and then a hierarchical multiagent system represents each mission. Using the semantic description of the mission plans, the agents reason about resources required for their missions and cooperatively decide on the resources that should be shared to carry out those missions. This is achieved at different levels of the agent hierarchy where policies and constraints are used during the decision process. Our experiments show that our approach leads to a better utilization of the resources and significantly improves the number of achievable missions when the number of available resources is limited.

1 Introduction

The word *mission* can be defined as a special duty given to a person, group, or an organization to carry out. In a broad sense, many goal-oriented activities in real life can be considered a mission, such as starting an enterprise, rescuing people after a flood and so on. Even though missions can be very different, they require critical resources to reach their final goals. Improper selection of the resources to be used for a mission may lead to its definite failure. Therefore, determining the resources that should be used to perform a mission is crucial, but not trivial. In real life, missions are planned by experts and the resources that should be used by a mission is determined after careful analysis of the mission's requirements. This means that considerable effort and expertise are required to determine the resources to be used by a mission.

Some missions may be critical and shall be given higher priority. For example, any failure in the mission of rescuing people after a flood may have severe consequences. Many organizations (e.g., United Nations, Red Cross, and so on) conduct hundreds of critical missions each year. These organization may require valuable but scarce assets

* Research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

such as aerial vehicles (e.g., helicopters), and sensors (e.g., radars) to carry out their missions. Therefore, they usually tend to share their assets to enable their missions to be carried out successfully. At different levels, this leads to the necessity of interoperability between different organizations to achieve their missions properly.

In this paper, we propose a framework where different organizations describe their missions semantically using an ontology. In this framework, each mission is divided into different operations, which are further divided into different tasks. All elements of a mission are related to each other with temporal and logical relationships, which are borrowed from OWL-S process ontology. This paper proposes a multiagent system to represent each mission so that the agents cooperate to determine the resources that can be shared among different tasks, operations and missions. We consider only the missions that are related to Intelligence, Surveillance, Target Acquisition and Reconnaissance domain. We select this domain to formulate and test our approach, because of the existence of well-established ontologies and standards for describing resources in this domain. However, our approach can easily be adapted to other domains as well by replacing the used ontologies. We experimentally show that our approach leads to a better utilization of the resources and significantly improves the number of achievable tasks when the number of resources is limited.

The rest of the paper is organized as follows. In Section 2, we describe how missions are planned and represented semantically. In Section 3, we describe how semantic matchmaking can be used to discover types of resources that can be used by a task. In Section 4, we describe the proposed approach for associating a multiagent system to each mission and cooperatively deciding on the resources that will be used by these missions. In Section 5, we evaluate our approach and Section 6 presents a discussion.

2 Mission Planning

Mission planning is the process of composing a plan containing the required steps in order to fulfill a mission. This plan should explicitly describe which operations are required by the mission. Each operation may need to be carried out following a specific schedule in order to achieve the goals of the mission. Hence, temporal relationships between operations are essential in the context of mission plans.

In addition to temporal relationships, there may be logical relationships between the operations while describing a mission plan. For example, we may need to specify that an operation should only be run if a specific condition is detected in the field. Hence, a representation of mission plans should be flexible enough to accommodate temporal and assorted logical relationships among operations. Below, we enumerate nine constructors used to define temporal and logical relationships between the operations of a plan:

1. **Sequence:** Defines a list of operations that need to be executed in order.
2. **Split:** Defines a set of operations that should be executed concurrently.
3. **Split+Join:** Defines operations that can only be executed after a set of concurrent operations are accomplished. It defines a kind of synchronization point in the mission plan.
4. **Any-Order:** Defines a list of operations that can be done in any order.
5. **Choice:** Defines a set of operations from which one should be chosen and executed.
6. **If-Then-Else:** Defines two operations so that one of them should be chosen and executed if a predefined condition holds; otherwise the other operation is executed.
7. **Iterate:** Defines an iteration over operation executions.

8. **Repeat-While:** Defines an iteration that continues while a predefined condition holds.
9. **Repeat-Until:** Defines an iteration that continues until a predefined condition holds.

Example 1 describes a mission that aims to provide relief to people injured in a natural disaster in an area where a civil war threat the humanitarian efforts. In order to fulfill this mission, a mission plan is prepared. This plan is depicted in Figure 1.

Example 1 We consider a hypothetical country in which there is an ongoing civil war between ethnical groups. An earthquake has recently hit and destroyed the eastern-most part of the country where thousands of civilians lost their homes and some of them are dead or seriously injured. This region is surrounded by mountains, where caves in the terrain are used by armed groups as shelters. Humanitarian volunteer organizations such as the Red Cross are willing to help the civilians. However, possible attacks by the armed groups in the region prevent these organizations from coming and operating in the region. In this setting, our mission is to enable these organizations to provide relief to those affected by the earthquake in the region without facing any security issue.

In the plan represented graphically in Figure 1, first intelligence is collected. As a result of this operation, the most appropriate place for building a camp for the injured civilians is determined. If the camp region is not clear, a cleaning operation is initiated. After ensuring that the region is clear, a camp and the required logistic channels are built using two different operations. These operations can be executed in any order. This means that the camp can be built in parallel with, before or after building logistic channels. This decision of the mission planner provides some flexibility on the execution of the plan. That is, if resources are scare, these two operations can be executed sequentially to make use of the same resources; otherwise they can be executed in parallel to decrease the overall mission completion time. Just after building a camp and setting up the logistic channels, a surveillance operation is initiated to protect the camp, roads to the camp and so on. In parallel, humanitarian organizations are escorted and civilians are moved to the camp. After civilians and volunteer organizations are in the camp, we start searching for civilians in need and escort these civilians to the camp iteratively until the end of the mission.

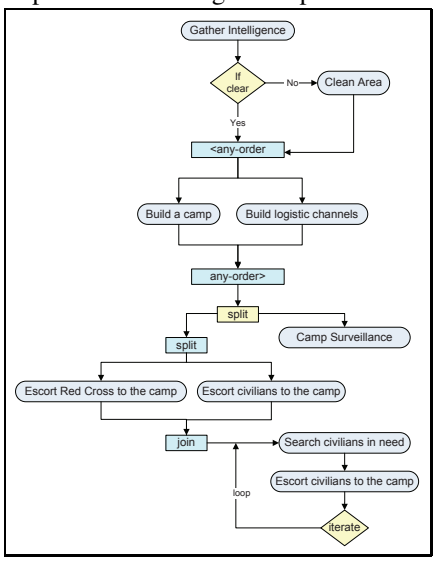

Fig. 1: A mission plan for Example 1.

Each operation is composed of tasks that are required to fulfill the objectives of the operation. Tasks are atomic and cannot be further divided into subtasks. A successful execution of an operation may require an ordered execution of tasks. Hence, within the context of a specific operation, we may clearly define when and under which conditions a task should be performed to achieve the objectives of the operation.

2.1 ISTAR Ontology

Missions are described using the concepts from an ontology that is specifically engineered for Intelligence, Surveillance, Target Acquisition and Reconnaissance (ISTAR) domain. we call this ontology as the *ISTAR ontology*. Figure 2 summarizes the main

concepts and relationships from this ontology. As shown in Figure 2, a mission may comprise several operations and in turn each operation may comprise several tasks. Therefore, tasks are located at the lowest level in the hierarchy. They specify how operations will be fulfilled and in turn operations define how the missions will be accomplished. Each task may require capabilities to achieve its objectives and assets may provide various capabilities. Assets provide the capabilities that are required by the tasks. *Platform* and *System* concepts are both assets, but systems can be attached to platforms. Sensors are regarded in the ontology as a specialization of systems.

In our approach, the ISTAR ontology of Figure 2 is represented using Web Ontology Language (OWL). Note that the ISTAR ontology shown in Figure 2 contains only core concepts and relationships. However, it is easily extended by adding other OWL ontologies to further elaborate different concepts. For example, an OWL-based sensor ontology composed of hundreds of sensor types and instances is simply added to the ISTAR ontology, so information about the capabilities provided by a variety of sensors is easily added to the ISTAR ontology. Similarly, individual ontologies for other concepts (e.g., *Platform*, *Task*, *Operation* etc.) are added to the ISTAR ontology to enhance it. As stated before, tasks require different capabilities to achieve their objectives. Capability requirements of tasks are divided into two categories:

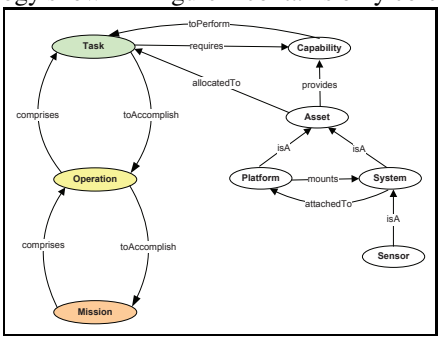

Fig. 2: ISTAR ontology.

1. **Operational requirements:** These are qualitative and quantitative parameters that specify the desired capabilities for a task within a context of a specific mission, or operation. These requirements are related to how the task should be executed in order to achieve desired operational effectiveness and suitability. For example, a road surveillance task for the mission in Figure 1 requires a high altitude capability as one of its operational capabilities, because this task will be held in a mountainous area. If we remove this operational requirement from the task, the task may not achieve its objectives for this specific mission.
2. **Intelligence requirements:** These requirements refer to the kinds of intelligence discipline (e.g., Radar Intelligence (RADINT), Imagery Intelligence (IMINT), and so on) that are required for a task. These requirements are used to select the sensors that can support such kind of intelligence. For example, if the road surveillance task requires IMINT intelligence, then cameras providing IMINT capabilities can be used for this task.

Operational and intelligence requirements can be associated with a task in three ways. First, a task can be defined abstractly in an ontology together with its default operational and intelligence requirements. Second, new requirements can be explicitly placed onto the task during the planning of a specific mission. Third, constraints defined within the scope of the mission may add new requirements to the task, or modify its existing requirements. Figure 3 shows how *Road Surveillance* task is defined abstractly in an ontology. This task has two requirements: constant surveillance (operational requirement) and IMINT capability (intelligence requirement). If we assume that

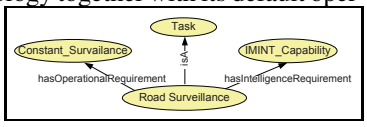

Fig. 3: Abstract task example.

the road surveillance task of the mission in Figure 1 is an instance of this task, then it has these two requirements by default. During the planning phase of the mission in Example 1, the IMINT capability of road surveillance task can be specialized further by adding PHOTOINT as its intelligence requirement, where PHOTOINT is a subconcept of IMINT and refers to "Photographic Intelligence".

Constraints imposed on the road surveillance task affects its requirements as follows. First, a *high altitude* operational requirement is added, because this task will be executed in a mountainous area. Second, because the road surveillance will be carried out during the winter (when snow, rain and fog are highly probable), *Radar Intelligence* (RADINT) is added to the intelligence requirements of the task. Figure 4 shows the resulting road surveillance task instance with its requirements. As explained above, constraints may affect the requirements of a task. For this purpose, we may use rules that represent the relationships between the constraints (e.g., terrain and weather conditions) and requirements (e.g., high altitude and radar intelligence). These rules are important because they capture the crucial domain knowledge.

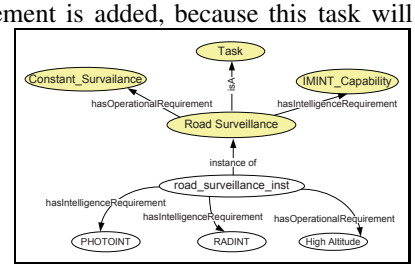

Fig. 4: Task instance example.

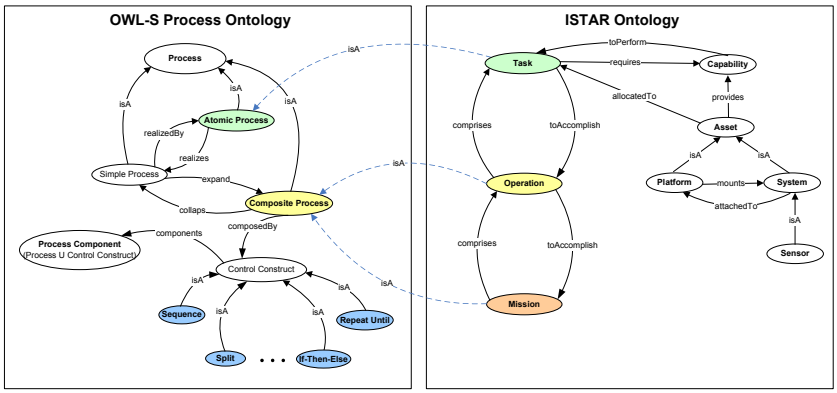

Fig. 5. Combination of OWL-S's process ontology and ISTAR ontology.

2.2 Describing Mission Plans

At the end of mission planning, a description of a mission plan should be produced. This description defines what steps are required to fulfill the mission in a timely manner. The ISTAR ontology provides fundamental concepts and relationships required to describe missions, operations, tasks, and so on. This ontology is written using a standard ontology language *OWL* [1]. Therefore, different software agents can easily understand, interpret and reason about the descriptions written using this ontology. However, the ISTAR ontology does not contain concepts and properties to describe temporal and logical relationships between operations, or tasks within a mission. Therefore, a mission plan that contains these relationships cannot be represented solely by the ISTAR ontology. In order to represent plans like the one in Figure 1, we need another ontology that has constructs to represent temporal and logical relationships between different entities.

OWL-S's process ontology[1] defines temporal and logical relationships between services (processes). These relationships are the same as the relationships we have described before. OWL-S defines three categories of processes: simple, atomic and composite processes. Composite processes are described using atomic or other composite processes using the aforementioned temporal and logical relationships (e.g., sequence, split, if-then-else and so on). In our context, tasks can be considered as atomic processes that accomplish sub-goals of operations and missions. Similarly, operations can be considered as composite processes that are composed of tasks. Lastly, missions can also be formalized as composite processes that are achieved by a set of operations. Therefore, we can combine OWL-S's process ontology and the ISTAR ontology in an intuitive manner as shown in Figure 5. In this way, missions, operations and tasks can be described iteratively with a standard formal semantics.

3 Semantic Matchmaking of Assets to Tasks

A mission plan contains operations and operations consist of tasks. Tasks require some operational or intelligence capabilities. Similarly, sensors (e.g., IR cameras) and platforms (e.g., Aerial Vehicles) have some capabilities that correspond to operational and intelligence requirements of tasks. Therefore, there is a strong correspondence between assets (sensors and platforms) and tasks. The purpose of this section is to exploit this correspondence to determine types of assets necessary to carry out tasks related to the operations of a mission. Once the necessary asset types are determined, concrete resources (instances of the determined assets) can be allocated to the specific tasks.

We assume that for each task there should be one platform that satisfies all its operational requirements. If a platform does not meet every operational requirements of a task, it cannot be used for the task. For example, an *Unmanned Aerial Vehicle (UAV)* that does not have high altitude capability cannot be used for the road surveillance task of the mission in Figure 1.

We propose a matchmaking algorithm, which depends on the semantic description of desired platforms and sensors required to accomplish a task. For this purpose, we define configuration of assets that can be used to achieve a specific task as follows:

1. **Adequate Platform:** For a specific task t, a platform p is adequate if it meets every operational requirements of t. It is defined as follows: $task(?t) \wedge platform(?p) \wedge \forall ?c(hasOpReq(?t,?c) \rightarrow hasCapability(?p,?c)) \Rightarrow adequate(?t,?p)$
2. **Deployable Platform:** For a specific task t, a platform is deployable if it is adequate and it can mount sensors that meet every intelligence requirement of t. It is defined as follows: $task(?t) \wedge platform(?p) \wedge adequate(?t,?p) \wedge \forall ?c\ (hasIntelReq(?t,?c) \rightarrow \exists ?s \wedge sensor(?s) \wedge mount(?p,?s) \wedge hasCapability(?s,?c)) \Rightarrow deployable(?t,?p)$
3. **Deployable Configuration:** For a task t, a deployable configuration is a set that contains a deployable platform p and a bundle of sensors that can be mounted by p, where their capabilities meet every intelligent requirement of t.

In this work, we use SPARQL [1], which is an RDF query language, with Pellet reasoner [2] to list deployable configurations for a given task instance. Figure 6 demon-

[1] http://www.daml.org/services/owl-s/1.1/Process.owl

strates an example SPARQL query for getting deployable configurations for the road surveillance task instance in Figure 4. This query returns a list of tuples where each tuple is composed of one deployable platform and a list of sensors that are attachable to the platform so that these sensors meet every intelligence requirement of the road surveillance task instance together. Therefore, each of the returned tuples corresponds to one deployable configuration.

```
PREFIX istar: <http://www.csd.abdn.ac.uk/ita/istar#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?p ?s0 ?s1
WHERE {
 ?p  rdf:type istar:Platform.
 ?s0 rdf:type istar:Sensor.
 ?s1 rdf:type istar:Sensor.
 ?p  istar:mount ?s0.
 ?p  istar:mount ?s1.
 ?p  istar:hasCapability istar:Constant_Surveillance.
 ?p  istar:hasCapability istar:High_Altitude.
 ?s0 istar:hasCapability istar:PHOTOINT.
 ?s1 istar:hasCapability istar:RADINT.
}
```

Fig. 6: SPARQL query example.

4 Selection of Resources for Missions using Agents

In Section 3, we describe how we can list all possible deployable configurations for a task using a reasoner and a query language. We should select one of these configurations depending on a utility function for the task. Then, for the execution of the task, we need to allocate specific instances of the assets referred in the chosen configuration. Note that instances of assets are scarce. This means that even though a specific configuration is best for the task, this configuration should not be chosen, because some of the assets in the configuration may not have enough instances available for the task. Scarcity of resources leads to a competition between tasks to acquire the instances of the desired assets. In this competition, a higher priority task has the privilege to acquire resources currently hold by lower priority tasks. Even if a task holds a specific instance of an asset, it may be taken away before the completion of the task if it is needed by a higher priority task and there is not another instance of the same asset available at the moment.

In this setting, proactive selection of the right configuration for each task becomes crucial. We should select a deployable configuration for each task so that the instances of these resources can be attainable for the tasks and the attained instances can be sustainable until the completion of these tasks without any prevention by the higher priority task. In order to determine the most proper deployable configurations for each task proactively, we propose to represent each mission plan as a virtual organization composed of agents representing tasks, operations and the mission. In the proposed approach, agents representing tasks communicate with other agents representing higher priority task to reveal the probability of sharing their resources. This way, agents cooperate to share their resources as much as possible, not only for better utilization of the resources but also for decreasing the possibility of intervention by higher priority tasks.

4.1 Proposed Multiagent Architecture

In the proposed approach, for each mission, we construct a virtual organization. Figure 7 shows how these virtual organizations look like. Each virtual organization has three levels: task level, operation level and mission level. At the task level, agents are responsible for finding the best solution in order to fulfill the task they represent. Therefore, they are knowledgeable about the details of the task they represent and each of them may have its own utility function and metrics of success. At the operation level, an agent is knowledgeable about the utility function, requirements and constraints of

the corresponding operation, as well as policies regarding the operation and the relationships between the tasks within the operation. Lastly, at mission level, an agent is knowledgeable about the success measure of the mission, policies related to the mission, relationships between the operations within the mission and constraints of the mission. Constraints include date, region, terrain and so on.

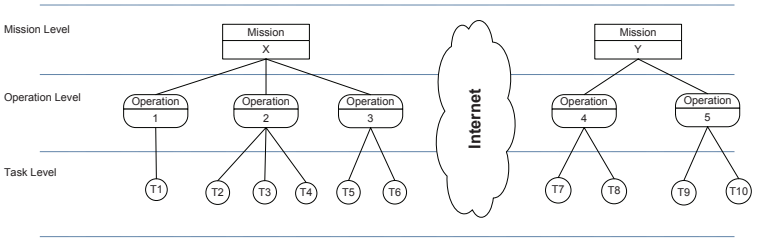

Fig. 7. Virtual organizations derived from two toy missions.

Operations within the same mission may have an opportunity to share their resources, given the fact that they are bounded by the same constraints at the mission level (e.g., region, terrain properties and so on). Example 2 provides a simple example of this type of resource sharing. However, this may not be the case for the operations belonging to different missions (i.e., they may have different regions, dates and so on). Similarly, tasks belonging to the same operation may have a better opportunity for sharing their resources with respect to other tasks that belongs to other operations or missions. That is, tasks within the same operation are bounded by the same constraints of the operation, so they may share their resources if their requirements are similar. Example 3 provides a simple example for such a resource sharing. On the other hand, it is harder for two tasks to share their resources if they belong to different operations, because each operation may have different constraints that affect the tasks within the operation (e.g., tasks in different regions or on different dates may not share their resources). Therefore it becomes a challenge to find different missions or operations so that the tasks within those missions or operations can share their resources.

Example 2 The mission in Figure 1 has two consecutive operations that are executed iteratively one after another: *Search civilians in need* and *Escort civilians to the camp*. Let us assume that both of these operations have similar operational requirements. This means that these two operations may use the same platforms for their tasks. This leads to an opportunity for using the same instances of assets for these consecutive operations. If using fewer asset instances has a better utility for the mission, the mission agent may request the agents of these two operations to cooperate by using the same instances.

Example 3 Let us assume that there is an operation called *detect and rescue civilians*, which is further divided into two tasks; *detecting civilians in need* and *rescuing civilians*. The best deployable configuration for the first task contains *Global Hawk* as a platform and the best deployable configuration for the second task contains *Black Hawk* as its platform. Therefore this operation requires two platforms *Global Hawk* and *Black Hawk*. However, using the same platform for both of the tasks has a higher overall utility for the operation. Hence both agents representing these tasks compromise and decide to use a *Pave Hawk* as platform, even if it is not the best solution for any of these tasks. Then, the sensors required for detecting and rescuing civilians in need are attached to this single platform.

4.2 Choosing Deployable Configurations to Share Assets

In order to determine which tasks should share their assets during their execution, we propose a protocol to be used with the multiagent architecture proposed in Section 4.1. This protocol consists of four main steps illustrated in Figure 8 and explained below:

1. A mission agent X, willing to share assets with other missions, posts a message on *Missions' Message Board*. This message contains the constraints that the mission X

has (e.g., the region and date of the mission), as well as the priority of the mission. Missions are willing to share their resources with higher priority missions. This is intuitive, because if a low priority mission X with a priority *level 1* shares an asset A with a higher priority mission Y with priority *level 5*, then the asset A cannot be taken away by other missions having priority lower than *level 5*.

2. To the message of the mission agent X, a set of other mission agents may respond if their constraints comply with the constraints within the message and their policies do not prevent them from sharing their assets with the mission X. For example, policies of a mission may not allow sharing assets with another mission unless these two missions belong to the same authority. Mission agent X creates a *Rendezvous Point* and invites responding mission agents to the rendezvous point, considering its own policies too. Through the rendezvous point, operation agents belonging to these missions can multicast IP messages among them and communicate to elaborate the possibility of sharing their assets. In our example of Figure 8, mission agent X creates a rendezvous point and invites mission agent Y. This means that operation agents of mission X and mission Y communicate through the rendezvous point. Note that, if no mission agent responds to the message of the mission agent X, it also creates a rendezvous point, but this time only its own operations communicate with one another through this rendezvous point to elaborate the possibility of sharing among them as in Example 2.
3. Operations that have similar constraints and requirements create sets. While creating these sets, policies related to these operations are also taken into account. A set may contain one or more operations. Then, for each set, a rendezvous point is created. Using this rendezvous point, tasks under the operations within the set communicate to decide on the assets that they can share. In Figure 8, we only show the set composed of $Operation1$ and $Operation5$ for simplicity. The tasks within these operations communicate through the created rendezvous point.
4. Each task agent in the rendezvous point computes the deployable configurations for the related task using the approach of Section 3. Then, each task agent publishes its desired deployable configurations and let other task agents vote for them. An agent's vote for a specific deployable configuration is based on the utility of sharing the assets within the deployable configuration for the agent. Lastly, depending on the voting results, each task agent decides on a deployable configuration that will be used during its executions. That is, task agents select deployable configurations that enable them to share as many resources as possible with others.

5 Evaluation

To evaluate our approach, we have prepared a set of missions [2] using the ISTAR ontology as described in Section 2. This set includes 20 missions, 143 operations and 221 tasks. For each mission, we built a multiagent system as in Figure 7. Note that, for two task agents to cooperate, they should be adequate to cooperate; at task level, operation level or mission level, there should not be any reason to prevent these tasks from cooperating (i.e., policies or conflicting constraints may lead to inadequacy to cooperate).

[2] Available in XML format at `http://www.csd.abdn.ac.uk/~murat/experiment.xml`

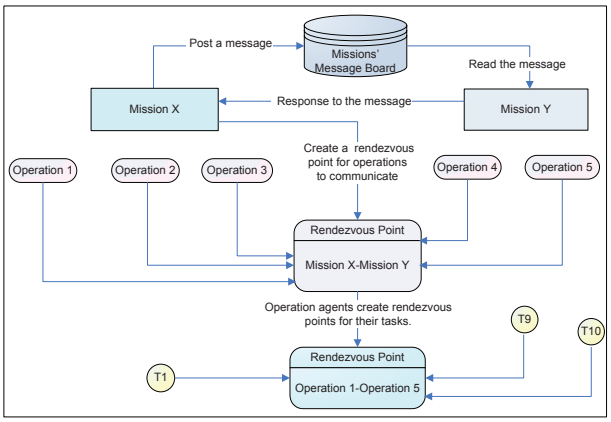

Fig. 8. Demonstration of the approach.

This leads us to determine a parameter in our evaluations, *ratio of tasks adequate for cooperation* (R_{ac}). For instance, $R_{ac} = 0.0$ means that policies or conflicting constraints (location and date of tasks) do not allow a task to cooperate or share its resources with other tasks in the system. On the other hand, $R_{ac} = 1.0$ means that policies and the constraints are created in mission, operation and task levels so that each task is adequate to cooperate or share its resources with other tasks in the system. Note that R_{ac} implies only the adequacy to share resources, but not the actual degree of resource sharing. That is, if two tasks require different resources, they cannot share their resources even though they are adequate to cooperate in terms of their policies or constraints.

In order to compare our approach, we also implement a naive approach to select resources for the tasks. In this approach, for each task, we individually select the best resources according to the requirements and the constraints of the task. Unlike our approach, this approach does not consider the cooperation or resource sharing between the tasks. Hence, each resource is allocated to one task. Our experiments show that this leads to 725 different resources (221 platforms and 504 sensors) having allocated.

Figure 9 shows the total number of required resources for different values of R_{ac}, when our approach is used. When $R_{ac} = 0.0$, agents are not adequate to cooperate, so the performance of our approach is same as that of the naive approach; the tasks require 221 platform instances and 504 sensor instances. However, when we increase R_{ac}, our approach causes more tasks to come together and search for possible ways of sharing their resources. This leads to a dramatic decrease in the required number of resources to carry out the missions. When $R_{ac} = 1.0$, our approach determine the resources that can be shared among as many task as possible, so the number of required resources decreases to 54 (17 platform instances and 37 sensor instances).

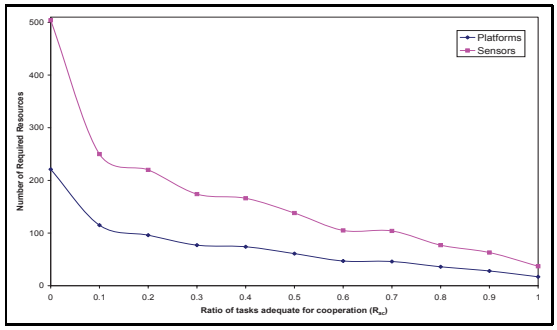

Fig. 9: Required resources vs. R_{ac}.

If there is not enough resources, a task cannot be executed. This may lead to

failure of an operation, which may further result in a failed mission. Therefore, in the next step of our evaluations, we measure how our approach improves the ratio of executable tasks when the number of available resources is limited. Figure 10 demonstrates the ratio of executable tasks for different R_{ac} values, while the number of resources ranges between 0 and 320. When tasks are not adequate for cooperation, only 44% of the task can be executed with all of the 320 available resources ($R_{ac} = 0.0$). For higher values of R_{ac}, our approach enables all of the tasks to be executed with fewer resources. This is expected, since our approach enable tasks to discover their opportunities to share resources, so more tasks are executed with fewer resources.

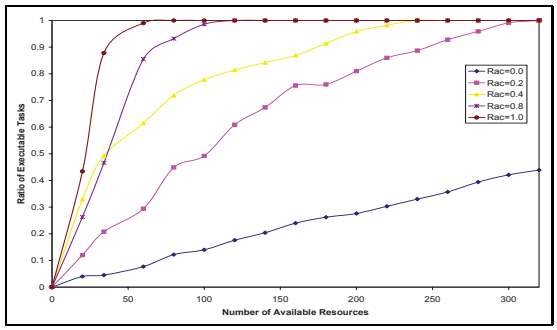

Fig. 10: Ratio of executable tasks vs. number of resources.

6 Discussion and Related Work

In this paper, we formulate a way of semantically representing plans to achieve missions. This representation enables us to describe the components of a mission plan semantically in terms of their requirements and relationships between them. Hence, we can reason about the resources necessary to carry out a mission as planned. Based on the proposed semantic representation of the mission plans, we propose a multiagent approach for the determination of the most convenient resources to carry out missions. We show that using this approach, missions can be carried out with fewer resources.

Policies define a general attitude or approach which can be described in terms of norms. For instance, the policy "resource x is expensive and its use should be regulated" could give rise to norms being put in place, defining who is allowed or prohibited to use x. Alternatively, policies can be seen as rules (as in, for instance, [3]) defining when norms (that is, permissions, prohibitions and obligations) are put in place or are revoked. Norms and policies have been used in disparate fields, ranging from security models of programming languages such as Java to the management of resources in distributed systems [3]. Explicit norms and policies, as opposed to implicit ones embedded in software, define computational behaviors in a flexible way [4].

Our approach supports policies and norms at different levels of abstraction (mission level, operation level and task level). It also enables missions from different organizations to share their resources without revealing the details of their mission plans. Cooperation policies of different organization may be different, so each mission considers its own policies and norms while choosing other missions for cooperation.

Different approaches are proposed before to select resources (sensors) for tasks. Many researches proposed utility-based solution. Johnson *et al.* propose to use energy conservation while selecting resources for tasks [5], whereas Bar-Noy *et al.* propose to select resources depending on the competition between the tasks [6]. In both case, the utility of a resource to a task is based on the geographical distance between them.

Therefore, in an essence all resources are of the same type and any resource can provide some utility to any task. We argue, this is not the general case and the reality is the contrary. Resources are heterogeneous (different capabilities, operational conditions etc.) by nature. Therefore, not all resources could provide even some utility to a task.

Various ontologies are proposed in the literature to describe sensors, platforms and so on. SensorML [7], OntoSensor [8] and Marine Metadata Interoperability project [9] are well-known examples. None of these approaches provides a high level representation to describe mission plans as a whole. Mission plans introduced in this paper can be considered as workflows, whose each component is described semantically using an ontology. In the literature, there are approaches that describe workflows semantically using an ontology [10]. However, these approaches do not consider multiagent systems to cooperatively select resources to carry out workflows as we do in this paper.

The experiments in Section 5 show that our approach is promising and deserves further research. Therefore, we are planning to extend our approach as follows. First, we want to generalize our approach by replacing the ISTAR ontology with a more generic ontology that will enable mission plans for various domains to be described easily (e.g., business missions such as starting an enterprise). Second, we want to extend our approach so that agents can create explanations for their choice of deployable configurations. Hence, human participants of a mission can understand the rationale behind a specific choice of resources for the mission.

References

1. Antoniou, G., Harmelen, F.v.: A Semantic Web Primer, 2nd Edition (Cooperative Information Systems). The MIT Press (2008)
2. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Web Semant. **5**(2) (2007) 51–53
3. Moffett, J., Sloman, M.: Policy Conflict Analysis in Distributed Systems Management. Journal of Organizational Computing (1993)
4. Vasconcelos, W.W.: Norm Verification and Analysis of Electronic Institutions. Volume 3476 of LNAI. Springer-Verlag (2004)
5. Johnson, M.P., Rowaihy, H., Pizzocaroz, D., Bar-Noy, A., Chalmers, S., Porta, T.L., Preece, A.: Frugal sensor assignment. In: 4th IEEE International Conference on Distributed Computing in Sensor Systems. (June 2008)
6. Bar-Noy, A., Brown, T., Porta, T.L.: Assigning sensors to competing missions. In: Proceedings of the Globecom2008. (November 2008)
7. Robin, A., Havens, S., Cox, S., Ricker, J., Lake, R., Niedzwiadek, H.: OpenGIS© sensor model language (SensorML) implementation specification. Technical report, Open Geospatial Consortium Inc (2006)
8. Russomanno, D.J., Kothari, C.R., Thomas, O.A.: Building a sensor ontology: A practical approach leveraging iso and ogc models. In: Proceedings of the 2005 International Conference on Artificial Intelligence (ICAI). (2005) 637–643
9. Bermudez, L., Graybeal, J., Arko, R.: A marine platforms ontology: Experiences and lessons. In: Proceedings of the ISWC 2006 Workshop on Semantic Sensor Networks. (2006)
10. Wang, Y., Cao, J., Li, M.: Goal-driven semantic description and query for grid workflow. In: SKG '07: Proceedings of the Third International Conference on Semantics, Knowledge and Grid, Washington, DC, USA, IEEE Computer Society (2007) 598–599

Trust Evaluation for Reliable Electronic Transactions between Business Partners

Joana Urbano, Ana Paula Rocha, Eugenio Oliveira

Faculdade de Engenharia da Universidade do Porto – DEI-LIACC
Rua Dr. Roberto Frias, 4200-465 Porto, Portugal
{joana.urbano, arocha, eco}@fe.up.pt

In the digital economy era, commercial relationships between business partners are increasing in flexibility, and temporary business binds tend to be created whenever a business opportunity arises. Moreover, the instability in demand increases the need for enterprises to procure new partners and the associate risk of inter-operating with partners that might be unknown beforehand. Therefore, enterprises need mechanisms that allow to evaluate the confidence they have on actual or potential partners, and to monitor this confidence in a continuous and automatic way. This paper evaluates a computational trust and reputation (CTR) system that provides estimated values of confidence on target partners. This system asymmetrically aggregates positive and negative evaluations of partners' behaviour, and introduces the percentage of successful contracts in the last *t* units of time as a first step to implement contextual factors in the partner's selection decision. The model was evaluated in an agent-mediated simulated textile virtual market, also described in this paper. We compare our approach with two other strategies of trust aggregation and present preliminary results that show that the asymmetric aggregation of evaluations and the introduction of the successful/violated contracts measure can improve the efficiency of the automatic selection of reliable partners in certain population scenarios.

1. Introduction

In the new era of digital economy, commercial relationships between business partners are increasing in flexibility, and business binds tend to be created whenever a business opportunity arises. The emergent need for new products and services, with increased quality, shorten time to market, and low price, and the instability in product demand, is forcing enterprises to risk new, sometimes unknown, suppliers, possibly spread all over the world. This new reality brings new technological, social, ethical, and economical challenges and risks to the industry.

Moreover, the desired automation of business inter-organizational relationships encounters some barriers in key stages, such as the selection of partners, negotiation and contract elaboration, particularly when there might be a large number of partners in the play (e.g. textile industry) that are unknown beforehand. The construction of reliable and broader accepted mechanisms of trust and reputation will allow organizations to continuously update their confidence level on actual and potential partners, in face of contextual changes. The benefits of such mechanisms are two-folded: i) allow

for a broader selection of partners, as it would be possible to infer confidence values for a major number of partners (both from reputation transmission and definition of contextual similarities/profiling); ii) make it safer for an organization to increment the degree of tasks that could be automated, both in the partner selection process and in the automated negotiation of contracts, based on trust and reputation mechanisms. These mechanisms are, in fact, getting great attention from different research areas, from social science and psychology to economics and computer science, particularly in the multi-agent and the service oriented architecture communities.

Our current research work focuses on the automation of inter-organizational interactions in two different but yet complementary tasks: the partners' selection and the negotiation of contracts in dynamic environments, taken as input confidence knowledge derived from trust and reputation. In this paper, we present our model of computational trust and reputation (CTR); particularly, we describe its aggregation engine that computes confidence values in a non-linear, asymmetric like way (i.e. confidence generally grows slower and declines faster), using properties of the shape of a hysteresis curve. In order to better understand the role of CTR systems in the process of selecting partners, we developed the simulated textile virtual marketplace (STexVM), a multi-agent system where buyers and suppliers of textile fabrics worldwide have the chance to conduct business. The aim of this system is three-folded: to study the dynamics of automated partner selection in an environment with different types of buyers and suppliers; to evaluate our model of aggregation of trust based on the shape of an hysteresis' curve, already implemented; and to evaluate the model of contextual fitness we are currently developing, which would bring organizational and business context as extra knowledge to the computation of confidence scores.

The structure of this document is as follows: section 1 introduces the paper and presents a brief revision on current research on trust and reputation. Section 2 presents STexVM, an agent-mediated simulated textile virtual market that we build to evaluate our approach. Section 3 presents our CTR system, and section 4 evaluates the model and compares it with two other different strategies. Finally, section 5 concludes the paper.

1.1 Literature Revision on Trust and Reputation

Current work on trust and reputation has diversified in multiple subfields. In the theoretical domain, there is important work on trust and reputation as elements of social intelligence. [1] addresses the theoretical issues related to reputation and image[1] in artificial societies and social simulation, and the authors extended recently their cognitive model of reputation in order to more thoroughly address the transmission of reputation ([2]). Probably the area where more research effort has been put, namely, in the multi-agents community, is the representation and aggregation of social evaluations into trust and/or reputation scores, which would serve as input to partner's selection in B2B scenarios. Some models have been proposed, from the simple eBay reputation

[1] The authors consider both image and reputation as social objects, as they concern shared information about the target "presumed attitude towards socially desirable behaviour". However, image is described as an evaluation belief, while reputation is a meta-belief, i.e., it is a belief about other agents' evaluations of a given target agent.

system that sums up integer values, to approaches that aggregates classifications using means and weighted means [3][4][5], Beta distributions [6], Dirichlet distributions [7], Bayesian approaches [8][9], and trust learning approaches [10] [11] [12]. Some of these models are implemented using complex beliefs, desires and intentions (BDI) architectures [5] [13]. A new trend of investigation on this area is the exploration of the business context to improve the decision making, raising significantly the number and type of information that the evaluator has in order to compute the trust. I.e., along with social evaluations given by direct experience or through witnesses, a plethora of new information related to the context of the business and of the organizations involved can improve the prediction of behavior of partners in a very significant way. However, few proposals have been made on this specific area [14], opening an enormous world of research.

2. The STexVM System

The STexVM is a simulated virtual marketplace for trading textile goods that ensures (as much as possible) reliable transactions, in a sense that it is able to detect business partners that in some moment start behaving in a defective way. The simulated environment is based on existent online virtual marketplaces where buyers and sellers in the textile and fashion industry can post buying and selling leads (e.g. the Fibre2Fashion marketplace).[2]

The STexVM follows the multi-agent paradigm, and is implemented over Jade platform[3], using the standard behaviours of Jade and FIPA[4] performative and interaction protocols. The key agents in this environment are buyers and suppliers. Buyers' agents represent companies that periodically need to buy a given amount of fabric (e.g. cotton, chiffon, voile) to textile suppliers. The type and quantity of fabric the buyer needs to purchase in a period of time (round period) are defined at its time of creation. At each negotiation round, a buyer can buy to one or more suppliers, until it reaches the defined quantity for the round. Based on these requirements, we can stipulate that a buyer agent has two main objectives: i) to periodically buy the needed quantity of the designated goods, in order to supply its operational activity; and ii) to maximize the utility it gets from the acquired material. In our simulated environment, the utility is related to the quantity and quality of the purchased goods it is able to buy at every negotiation round. Therefore, the choice of a reputable partner is essential to the lifecycle of the buyer.

Supplier agents represent textile companies that periodically need to sell a given amount of fabric (e.g. cotton, chiffon, voile) to buyers. Each supplier shall provide two different types of fabric, and the exact type and quantities of fabric the supplier needs to sell in a period of time (round period) are defined at its time of creation. The supplier agent was purposely designed to be simple, and its main objective is to periodically sell a determinate amount of the goods it has to sell, emulating a real world manufacturer and exporter of textile fabrics. The remaining agents of the STexVM

[2] http://www.fibre2fashion.com/
[3] http://jade.tilab.com/
[4] http://www.fipa.org/

system are the Agent Simulation Manager, who manages the configuration parameters related with buyers and suppliers; the Agent DF, which registers competences of buyers and suppliers; and the Agent CTR, which gathers information about the performance of suppliers and computes their confidence scores on-demand, when requested by the buyers. Figure 1 illustrates the relation between these agents.

2.1 Initial Configuration of Suppliers and Buyers

Each buyer and supplier that enters the simulated virtual marketplace gets a random configuration over a predefined set of values. In order to perform our simulations in a scenario that would approach a real textile environment, we picked and mangled real data from online virtual marketplaces.

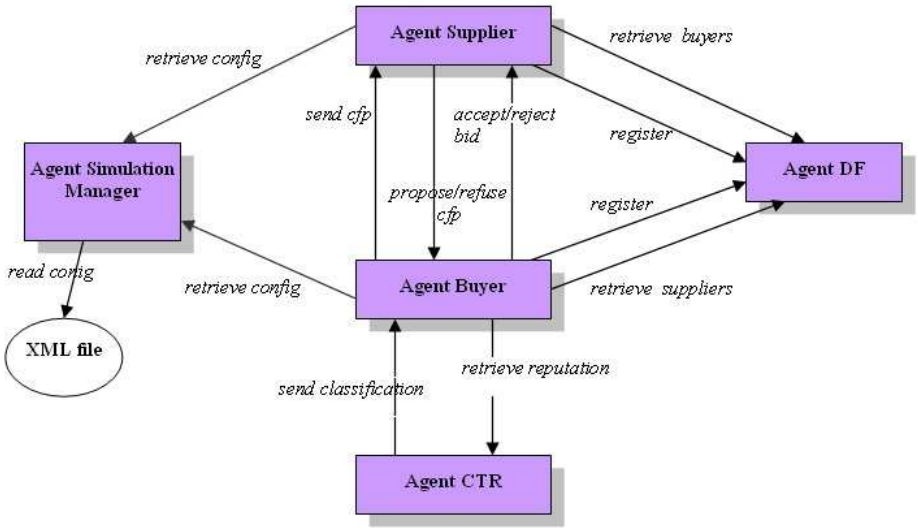

Fig. 1. Interactions between agents in the STexVM system

All suppliers in the STexM system assume the role of *manufacturer & exporter*, for simplicity. Suppliers are characterized by a set of properties that are setup when a new instance of a supplier is created; for each new supplier, two different fabrics are chosen from the values presented in Table 1 (left), and they are associated to a quantity randomly picked up from the values on the table (right). Each supplier also gets a country of registration, a year of establishment, and the total number of staff and annual sales. The latter three values are taken from real data collected from a virtual marketplace in the textile domain. Finally, each supplier is initially assigned an estimated "behaviour" from three possible values: "good", "fair" and "bad".

Buyers are characterized by their *buying characteristics*, related to the good they need to buy periodically (number and type) and respective range of quantities. Again, these values must be randomly picked up from the values in Table 1, every time a new instance of the buyer agent is created.

Table 1. Type of fabrics (left) and quantities (right) that can be transacted in the virtual marketplace

Type of Fabric (Good)	Quantities (meters)
Blended, Chiffon, Cotton, Crepe, Denim, Dyed, Embroidery, Fibre Waste, Fleece, Grey, Interlock, Jersey, Knitted, Lycra, Nylon, Polyester, Silk, Spandex, Terry Cloth, Velour, Velvet, Viscose, Voile, Wool, Woven	500; 5,000; 25,000; 50,000; 80,000; 120,000; 180,000; 240,000; 500,000; 1,000,000

2.2 Selection of Partners

At every negotiation round, each buyer issues a call for proposals (cfp) that sends to all registered sellers of the good in cause. A contract-net like negotiation occurs, and the buyer selects a number $n > 0$ of partners that optimizes the expected utility in the round, using equation (1).

$$E(u) = \arg \max_{i} for\ each\ i \sum_{j} util_j * trust_j \quad (1)$$

In the equation above, i stands for the possible combinations of suppliers' proposals that fit the quantity specified in the current cfp, not exceeding it; j represents the suppliers considered in each of these combinations, and $trust_j$ is the confidence score computed for supplier j at selection time Finally, $util_j$ is the quantity proposed by each supplier j in the round, normalized by the quantity specified in the cfp, i.e., $quant_j/Quant$. In our system, a buyer can order less quantity than the maximum quantity (*Quant*) defined in the cfp, but it cannot exceed *Quant*. Also, a buyer cannot accept partial quantities of the received bids.

3. The Proposed CTR Model

3.1 Aggregation using Non-Linear Curves

We developed an aggregation model that allows for the expression of non linearity in the process of trust constructing. On one hand, it captures an important feature of the dynamics of trust as defined in [1], the *asymmetry*, where trust is hard to gain and easy to lose. On the other hand, our model extends the non-linearity concept to distinct phases on the process of trust construction. In this context, Melaye and Demazeau (2005) address the asymmetry question in [15], but focus their approach on direct observations and do not consider different phases in the trust construction process. At the extent of our knowledge, none of the existent CTR models addresses the non-linearity of trust dynamics in a practical way.

In order to model the non-linear behaviour of our aggregation engine, we were intuitively influenced by the physical phenomena of hysteresis, and developed simple heuristics based on the hysteresis curve to pre-validate our assumptions, as described above. Figure 2 shows a hysteresis curve obtained from the application of Rostilav Lapshin formula [16] depicted in equation 2.

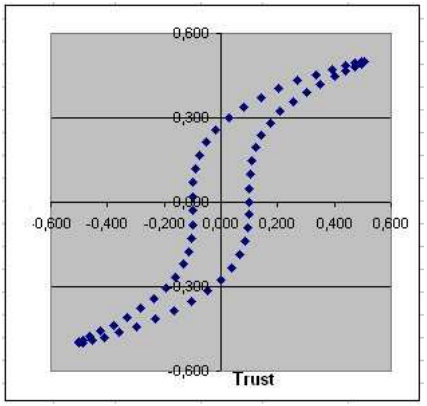

Fig. 2. The shape of a hysteresis curve with parameters a=0.1, b_x=0.5, b_y=0.5, n=1 and m=3. In the formula, *a* represents the coersitivity parameter, *m* and *n* are integers used to fit the curve and b_x and b_y are the saturation parameters.

$$x(\alpha) = a.\cos^m \alpha + b_x.\sin^n \alpha$$
$$y(\alpha) = b_y \sin \alpha \quad (2)$$

In our model, the trust of a given supplier is captured in the OX axe (see picture above). We use the rightmost plot of the curve to aggregate positive results (e.g. evaluations, contract results), and the leftmost plot to aggregate negative results. This signifies that the model captures sudden changes in the behaviour of the suppliers, and that the impact of negative results is weaker when the confidence value of the supplier is rather low or rather high, and stronger when the supplier is in the process of acquiring trust, following here common sense about confidence construction.

3.2 Contextual Fitness

Social evaluations help the selector to predict how well a given candidate partner will execute a task and to compare between several candidate partners. However, there are some questions a selector would pose before making a decision that cannot be answered by simply aggregating available classifications in the form of trust and reputation values. These questions involve somehow a certain level of *intuition*. As an exemplificative case, consider a high tech company that fears to select a partner from a country of origin without high technology tradition, even though this partner has proved high quality work in the desired task in the recent past. This is the type of contextual knowledge we intend to incorporate in our CTR model, and we name it *contextual fitness*, as it will provide extra decision information to the selector by computing a value of *how well the candidate partner fits in the selector current needs*, as defined in the issued cfp. Additionally, the selector can also provide additional preferential conditions, such as:

C1. *DeliveryTime ≥Price ≥ Quality;* ***C2.*** *OriginCountry !=p {countryA, countryB}*

In C1, the selector expresses an order of preference among attributes of the CFP, using the operator $\geq$, and in C2 the selector indicates that it preferentially would not deal with candidate partners for a list of origin of countries. The contextual fitness component, not yet formally described, is then an inference engine that analyses the *profile* of the actual business, supported by the cfp and by conditions defined by the selector, and the *profile* of the candidate partners, and infer how well they adapt. The profile of the partners is given taking into account historical relevant classifications stored at the CTR system and financial and organizational characteristics of the partners stored in a registry service. We use again an example to help clarifying the value of the contextual fitness component: let us imagine that an entity pretends to explore a business opportunity in the fashion market and creates a virtual enterprise for the fabrication of fashionable t-shirts. This entity starts to look for partners in the fashion/textile industry, including designers, garment manufacturers, fabrics suppliers, accessories' dealer, etc. Now, for the sake of simplicity, consider that the selector initiates the selection of partners' process and issues this (rather simplistic) description of component "zipper", included in the cfp:[5]

material: <cotton; nylon>; quantity: [10000-7500]; weight: <150; 200>; color: <red; purple>; delivery time: 2 weeks

Consider also that the selector has defined the following contextual condition:

C1. Delivery Time = ABSOLUTE

Now, let us imagine that a given candidate partner has proven to be reliable in the past in providing 150g red cotton zipper in two weeks, but he never traded more than 5000 units at a time. In this case, the inference engine can predict, based on the candidate profile, whether the candidate partner would be able to deliver such a quantity in the cfp terms. The mentioned profile can be build upon several parameters, such as the partner's tendency in negotiation over the time, organizational information (e.g. number of employees, country of origin, size of the industrial plants) and several financial figures.

4. Simulated Experiments

We developed a simple simulation scenario to evaluate the proposed non-linear model of trust aggregation and to compare it with other approaches. Therefore, we defined four different strategies to partner selection and tested each one of them in the STexVM system. The QUANT strategy orders candidate bids by decreasing quantity and then keeps selecting every proposal with quantity equal or less than the remaining quantity, until the quantity defined in the cfp is reached. This strategy does not take into consideration the trust values of the suppliers. In the remaining three strategies, the selection of partners is done taking into account the expected utility gain in select-

[5] In the above example, the tuples indicated the preferred order of values (e.g. the selector prefers zippers weighting 150g to 200g), and the first value in range parameters indicated the preference of the selector (e.g. he accepts to deal quantities from 7500 to 10000 units, although it prefers to buy as closed as the higher value possible).

ing one ore more suppliers in the current round, as specified in equation 1.[6] Therefore, strategy ASYM uses our non-linear aggregation engine to compute the trust component, while the WMEAN strategy uses an aggregation engine that computes the mean of the results weighted by the recency of the results (cf. [17]). As mentioned earlier, there are several CTR models that use weighted means to aggregate social evaluations, therefore the WMEAN strategy will allow us to compare our strategy with one that is disseminated in the trust and reputation community. Finally, strategy ASYM+ adds the percentage of successful contracts of the supplier in the last N transactions/units of time to the ASYM strategy, by multiplying this percentage with the confidence score returned by the ASYM aggregation engine.

For all the strategies, in the first rounds of each experiment the buyers start to explore the space of available candidate partners, by randomly selecting the partners, and after some rounds they progressively increase the exploitation by selecting partners based on the chosen strategy. The experiments are homogeneous, in the sense that in each simulation run we populated the simulated environment with buyers following the same strategy.

4.1 Experimental Methodology

In order to evaluate the four strategies described above, we used three different populations of suppliers' agents. We consider three types of behaviour for suppliers ("good", "fair", and "bad"), where the behaviour of a supplier is related to the results of the contracts it makes during it lifecycle with buyer agents. Each supplier was assigned a behaviour at its creation time, following a uniform distribution over the three possible values. We consider that the capacity of each type of suppliers in fulfilling the contract is modelled by a Markovian process with two states (1 and 0, standing for contract fulfilment and contract violation, respectively) and transition probabilities P11 (Fulfilment-to-Fulfilment) and P01 (Violation-to-Fulfilment). The values considered for populations A, B and C are defined at Table 2.

Table 2. Values of the transition probabilities used in the experiments. $P_0 = 0.50$ for all cases.

	Type "Good"		**Type "Fair"**		**Type "Bad"**	
	P11	P01	P11	P01	P11	P01
Popul. A	0.90	1.00	0.80	0.75	0.50	0.50
Popul. B	0.90	0.90	0.90	0.70	0.90	0.50
Popul. C	0.70	0.70	0.70	0.70	0.70	0.70

[6] We shall note that equation 1 takes into account three possible terms of inter-organizational business contracts, namely, the delivery of the contracted good, the partial delivery of it (e.g., supplier *x* succeeds to deliver the fabric but supplier *y* does not), and the quality of the supplied good, where 0 corresponds to an unacceptable quality good, and 1 corresponds to an excellent quality product; although we restrict the classification of the received value to a Boolean value, we intend it to be picked from a broader range of possible values, either discrete (e.g., labels as "good", "acceptable" and "not acceptable") or continuous (e.g. in the [0, 1] range), in a future version of the system.

As can be seen from the table above, in population A, suppliers of type "Good" have high probability of success and never fail two contracts in a row (once P00 is zero). The remaining types correspond to worse behaviours. In population B, all suppliers have the same, high probability of fulfilling a contract (P11 is 0.90), but suppliers of type "Good" are less prone to fail more than one contract in a row than the remaining types of suppliers, being "Bad" the worst case. In population C, all suppliers have the same (rather low) probability of fulfilling their obligations. Although the proposed population modelling does not directly mirror reality, it allows us to study how the different strategies respond to the resulting patterns of contract violations.

For each one of the four strategies defined above, we run 4 experiments per suppliers' population, in a total of 12 experiments or runs per strategy. Also, at each run, we launched 15 buyers and 75 suppliers, and each buyer issued 50 cfp at corresponding negotiating rounds. At this point, it is convenient to remind that a cfp discriminates the fabric and the quantity of material that the buyer intends to purchase, and it is sent to all suppliers that sell the desired fabric; in response, every supplier that receives a cfp sends a proposal with, at maximum, the required quantity of the fabric material, or refuses to propose if it not able to satisfy the cfp requirements in the current round. Also, each supplier is able to replenish its stock at the start of each negotiation round.

Finally, the utility gained by each buyer at each negotiation round was recorded, and at the end of the experiments the average utility of a buyer and the corresponding standard deviation were evaluated for each one of the considered strategies. The average utility captures the capacity of the buyer in selecting good suppliers, and, this way, allows for the evaluation of the performance of each one of the four strategies. Table 3 presents compact data about the experiments.

Table 3. Values and parameters used in the experiments

Quantities	5000, 50000, 180000, 240000, 500000, 1000000
Fabrics	Chiffon, Cotton, Denim, Dyed, Jersey, Fleece
# buyers	15
# of sellers	75
Types of sellers	Chosen upon a uniform distribution over the types {"good", "fair", "bad"}
# issued CFP per buyer, per run	50
# runs per strategy and per population	4
# past results (Hyst+ strategy)	8
Exploit/Exploration formula	Uniform distribution over f(x), where f(x) = 100 – $round_i$ * 7 or 10, if (100 – $round_i$ * 7 < 10)
Hysteresis parameters	a = 0.1, b_x = 0.5, b_y = 0.5, m=1, and n=3; the parameter α increases/decreases in steps of $\Pi/12$units.

4.2 Results and Conclusions

Figure 3 shows the average utility of buyer agents obtained per strategy and per population, in percentage. In population A, the ASYM strategy outperforms the others three, with buyers buying, in average, 83.81% of the desired good in the desired quan-

tity (ASYM+ reached 80.31%, QUANT 70.18% and WMEAN 79.67%). Concerning population B, the ASYM+ got 83.73% of average utility, outperforming the remaining three (ASYM: 81.81%, QUANT: 81.68%, and WMEAN: 82.27%). Finally, in population C, the obtained results were quite similar (ASYM: 67.63%, ASYM+: 68.21, QUANT: 68.72% and WMEAN: 68.17%).

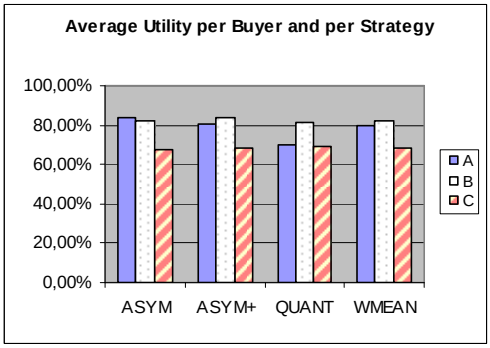

Fig. 3. Average utility per buyer, strategy and population (A, B and C)

While ASYM and ASYM+ performed a little better that WMEAN and QUANT, in reality we would expect that ASYM+ strategy would show greater advantage when compared to the others three strategies, because of its ability to asymmetrically aggregate trust and to take recency into account. Therefore, in order to understand the effect of random partner selection at the exploration rounds in the results, we traced the last 50 transactions of every run, for each strategy and for each population, and averaged the number of unsuccessful results (we remind here that at the last 50 transactions the probability of a random selection of partners is 10%). Figure 4 shows the results we obtained. In reality, we observed that the strategy ASYM+ outperformed the remaining strategies in population A (13% of violated contracts, against 14% of ASYM, 22.50% of QUANT and 15.50% in WMEAN) and in population B (9% of violated contracts, against 12.50% of ASYM, 16.50% of QUANT and 14% of WMEAN). Considering population C, the WMEAN strategy presented poorer results than the QUANT strategy (30% against 28.50%), and strategies ASYM and ASYM+ got an average of violated contracts of 24.50% and 25.50%, respectively.

In order to reduce the possibility of noise, we rerun the experiments for population A and strategies ASYM+ and WMEAN, considering 12 runs for each strategy. We observed that ASYM+ got an average of 12.17% of violated contracts in the last 50 transactions, outperforming once again WMEAN, which obtained 14.83% of contract violations. Also, we observed that 80.17% of the suppliers selected in the last 50 transactions were of type “Good”, for the ASYM+ strategy, while this number reduced to 69.83% for the WMEAN strategy.

Analysing this latter data, we can conclude that the non-linear aggregation of trust leads to a better estimate of the future behaviour of suppliers than the weighted mean with recency approach, allowing for the effective reduction of violated contracts. Moreover, the empirical analysis of the traces captured from the experiments shows that such a model that embeds the historical construction of trust is more effective in predicting patterns of behaviour than approaches that simply averages evaluations.

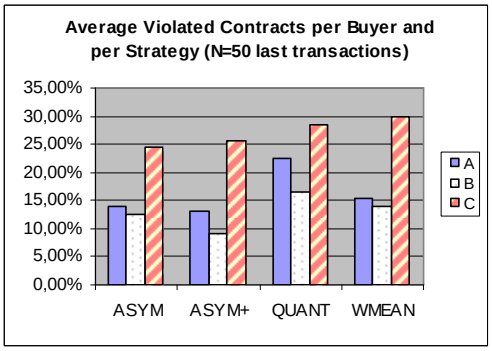

Fig. 4. Average number of violated contracts per buyer, strategy and population (last 50 transactions)

4. Conclusions and Future Work

This document presented the STexVM system, a simulated virtual marketplace that mirrors the posting of buying and selling leads of real textile virtual market, while adding up an important component of automation of the process of partner selection, essential to the interoperability concept. This system follows the multi-agent systems paradigm, and was implemented in Java over the Jade Platform. One important component of the developed prototype is it module of computational trust and reputation (CTR), an aggregation engine that computes confidence scores using a non-linear, hysteresis like approach.

We evaluated the performance of the non-linear strategy, measuring the number of successful contracts per buyer in the total number of cfp issued by the buyer in each experiment. Therefore, we measured the capacity of the buyer in selecting different types of suppliers based on the estimated trust. Then, we compared this approach with a trustless-based strategy that selects partners by the quantity they are available to supply, and to a strategy that estimates the trust of suppliers using a weighted average aggregator. The results obtained seem to validate our hypothesis that a non-linear strategy allows for a better detection of good suppliers, leading to a reduced number of broken contracts, when compared with strategies that aggregates evaluations based on average statistics. Based on these results, another formalization of non-linearity aside from the hysteresis approach might be explored. As future work, we will change the STexVM system in order to keep contractual information about suppliers between different experiments, avoiding the bootstrapping of the system and the associated noise derived from the initial strong exploration phase.

Acknowledgements

The first author enjoys a PhD grant with reference SFRH/BD/39070/2007 from the Portuguese Fundação para a Ciência e a Tecnologia.

References

1. Conte, R., M. Paolucci. 2002. *Reputation in Artificial Societies: Social Beliefs for Social Order*, Kluwer Academic Publishers: Dordrecht, 2002, ISBN 1402071868
2. Salvatore, A., I. Pinyol, M. Paolucci, J. Sabater-Mir. 2007. "Grounding Reputation Experiments. A Replication of a Simple Market with Image Exchange", in Proceedings of the Third International Model-to-Model Workshop, France, p.32-45 (2007)
3. Ramchurn, S., C. Sierra, L. Godo, N.R. Jennings. 2004. "Devising a trust model for multi-agent interactions using confidence and reputation", International Journal of Applied Artificial Intelligence (18) (2004) 833–852.
4. Sabater, J., C. Sierra. 2001. REGRET: A Reputation Model for Gregarious Societies. In *Fourth workshop on deception fraud and trust in agents societies*, 61-70
5. Carbo J., J. Molina, J. Davila, 2001. "A BDI Agent Architecture for Reasoning About Reputation", in *IEEE International Conference on Systems, Man, and Cybernetics*, v2, 817-822
6. Jøsang A., R. Ismail. 2002. "The Beta Reputation System", in *Proceedings of the 15th Bled Electronic Commerce Conference*, Slovenia
7. Reece, S., A. Rogers, S. Roberts, and N. R. Jennings. 2007 A Multi-Dimensional Trust Model for Heterogeneous Contract Observations. In: Twenty-Second AAAI Conference on Artificial Intelligence, July 2007, Vancouver, Canada.
8. Zacharia, G., P. Maes. 2000. Trust management through reputation mechanisms. In *Applied Artificial Intelligence*, 14(9), 881–908
9. Haller, J. 2007. A Bayesian Reputation System for Virtual Organizations, In *Dagstuhl Seminar Proceedings 06461, Negotiation and Market Engineering*
10. Hang, C., Wang, Y., and Singh, M. P. 2008. An adaptive probabilistic trust model and its evaluation. In Proceedings of the 7th international Joint Conference on Autonomous Agents and Multiagent Systems -V3 International Conference on Autonomous Agents. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1485-1488.
11. Erete, I., E. Ferguson, and S. Sen. 2008. Learning task-specific trust decisions. In Proceedings of the 7th international Joint Conference on Autonomous Agents and Multiagent Systems - Volume 3 International Conference on Autonomous Agents. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1477-1480
12. Smith M., M. desJardins. 2009. "Learning to trust in the Competece and Commitment of Agents", in Autonomous Agent Multi-Agent Systems (2009). 18:36-82
13. Sabater-Mir, J., I. Pinyol, D. Villatoro, G. Cuní. 2007. "Towards Hybrid Experiments on reputation mechanisms: BDI Agents and Humans in Electronic Institutions", in XII Conference of the Spanish Association for Artificial Intelligence (CAEPIA-07), V2, p.299-308
14. Rettinger, A., M. Nickles, and V. Tresp. 2008. A statistical relational model for trust learning. In *Proceedings of the 7th international Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*, Richland, SC, 763-770.
15. Melaye, D., Y. Demazeau. 2005. "Modèles et Réseaux de Confiance", in *Les Cahiers Leibniz*, n.º 142, Dec. 2005, ISSN: 1298-020X
16. Lapshin, R. V. 1995. Analytical model for the approximation of hysteresis loop and its application to the scanning tunneling microscope. In *Review of Scientific Instruments*, volume 66, number 9, pages 4718-4730, September 1995
17. Huynh, T. D., N. R. Jennings, N. R. Shadbolt. 2006. An integrated trust and reputation model for open multi-agent systems, in *Autonomous Agents and Multi-Agent Systems*, Vol. 13, N. 2, September 2006, pp. 119–154

An integration of a Semantically Enabled Service Oriented Architecture and Agent platforms

Ingo Zinnikus[1] and Srdjan Komazec[2] and Federico Facca[2]

[1] DFKI GmbH, Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
ingo.zinnikus@dfki.de
[2] STI Innsbruck, Technikerstrasse 21a, 6020 Innsbruck, Austria
{firstname.lastname}@sti2.at

Abstract. The capability to provide on-demand service access for SMEs can further reduce costs and allow companies to concentrate investments on their core businesses which in turn facilitates the overall competitive advantage. In the FP7 European Project COIN, a platform for supporting SMEs is developed which combines the flexibility of an execution environment for Semantic Web services with agent-based service compositions. One major benefit of the combination is that through this the potential of agent systems, most notably flexibility, can be tapped. An obstacle is the integration of dynamically discovered services which often cannot be used by agents because of interoperability problems (e.g. data heterogeneity). By delegating this task to WSMX, the agent platform can concentrate on coordination tasks. This paper presents an approach to integrate an implementation of the Semantically Enabled Service Oriented Architecture (SESA) with an agent platform. The integration approach is motivated and described in the context of a high-level scenario coming from the aeronautical industry.

1 Introduction

The last few years saw a growing need for a flexible and highly adaptive service provisioning solutions. While large enterprises have enough resources to deploy their own IT infrastructure and services, SMEs[3] require more flexible options. SMEs often cannot afford the cost of service deployment especially if those services are needed in a limited time or number of transactions. The capability to provide on-demand service access (when they are required and without the need to invest in the maintenance cost) can further reduce costs and allow companies to concentrate investments on their core businesses which in turn facilitates the overall competitive advantage. These are some of the most relevant foundations for the Software as a Service [7] paradigm.

In the FP7 European Project COIN[4], a platform for supporting SMEs is developed which combines the flexibility of an execution environment for Semantic

[3] abbr. Small and Medium Enterprises
[4] http://www.coin-ip.eu

Web services with agent-based service compositions. The main objective of the COIN Integrated Project is to allow industrial Networked Enterprises (supply chains, collaborative networks, business ecosystems) to access the potentials of the Future Internet and of the Internet of Services (IOS) in particular. This objective is achieved by means of a Generic Service Platform (in the following: *COIN platform*, i.e. the bridge between Enterprise Environments and Collaborative Platforms EE/CP on the one side and the IOS on the other side), which is a well known implementation of a SESA (i.e. WSMO [4]) enriched with security, pervasiveness-scalability and intelligence properties. In particular, intelligence is meant here as the capability of the COIN platform to properly interpret service requests (or *Goals* in the WSMO terminology) originated by EE/CP and to activate the service search-discovery-composition functions. In COIN, we are addressing this issue mainly at design time, by providing agent-based mechanism for an intelligent, business-driven decomposition of the goal and advanced negotiation capabilities for the definition of the relevant service level agreements.

The execution environment for Semantic Web services is based on the Web Service Execution Environment (WSMX). One major benefit of the combination of WSMX and agents is that through this the potential of agent systems, most notably flexibility, can be tapped. A notorious problem is the integration of dynamically discovered services which often cannot be used by agents because of interoperability problems (e.g. data heterogeneity). By delegating this task to WSMX, the agent platform can concentrate on coordination tasks.

In this paper, we describe the details of our approach for integration. The paper is structured as follows. In section 2, we describe the core components of the COIN platform. In section 3, we introduce the scenario which we use for illustration. The approach for service composition is presented in section 4. In section 5, we discuss open issues and problems and conclude in section 6.

2 The COIN Platform

The COIN Platform represents a service provisioning platform capable of supporting the SaaS-U paradigm for Enterprise Collaboration and Enterprise Interoperability services. The overall idea is to provide a set of services through the platform that enable SMEs and large enterprise to create virtual organizations leveraging collaborative services and exchanging information through integration services. All collaborative and interoperability services, according to the service agreements subscribed by the different stakeholders of the platform, are provided within a pay-per-use fashion (or other form of long term subscription to the platform). As presented in Figure 1 the platform is composed of the Web Service Execution Environment (WSMX[5]) as the foundation of the platform, the TrustCoM[6] security gateways which provide trust and security in a SOA implementation, a Peer-to-Peer repository/registry (coming from Digital Busi-

[5] http://www.w3.org/Submission/WSMX
[6] http://www.eu-trustcom.com

ness Ecosystems project[7]) which provides fail-safe storage facilities, and agents for intelligent service compositions and negotiations. This paper highlights the integration details of WSMX and agent platforms.

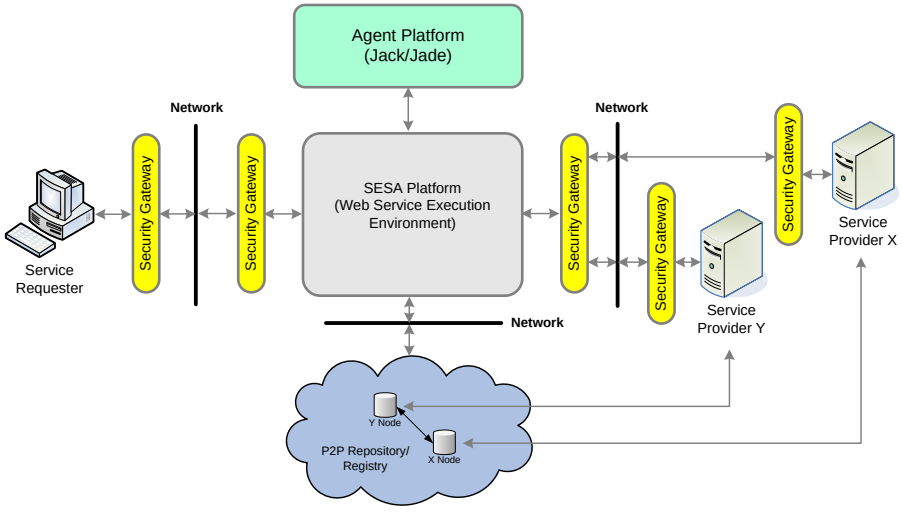

Fig. 1. The COIN Service Provisioning Platform

2.1 Semantically Enabled Service Oriented Architecture

The initial Web service technology stack allows exchange of messages between the parties by leveraging SOAP[8]. Furthermore, it allows for description of the technical interface for Web service consumption in the form of WSDL[9]. These technologies form the soil foundation for an implementation of the Service Oriented Architecture (SOA) paradigm that represents the dominant approach in employing service orientation in delivery of business functions. However, these technologies only support Web service usage by manual inspection and integration, i.e. existing SOA solutions are proving difficult to scale without a proper degree of automation [8]. Tasks such as service discovery, selection and ranking, composition, mediation, negotiation, and execution of Web services require that involved services are completely characterized by their interfaces. However, in traditional Web Service implementations, the lack of information to express the meaning of the data and of the vocabulary referenced by the interface as well as the lack of the formalization of the behavior is as a matter of fact prohibiting or at least hindering the automation of the envisioned tasks.

[7] http://www.digital-ecosystem.org
[8] http://www.w3.org/TR/2000/NOTE-SOAP-20000508
[9] http://www.w3.org/TR/wsdl

The concept of Semantic Web Services (SWS) [6] aims at providing more automated usage process of Web service by leveraging the Semantic Web [2]. SWS utilize ontologies as the underlying data model in order to support semantic interoperability between Web services and its clients and apply semantically enabled automated mechanisms that span the whole SWS lifecycle. More generally, the introduction of semantics as an extension of SOA and creation of Semantically Enabled Service Oriented Architectures [3], provides for next generation of service-oriented computing based on machine processable semantics.

In order to provide support for Semantic Web Services, two main approaches are envisioned. The former one (the bottom-up) approach relies on changing and extending existing models of Web services with the support for explicit semantics. This approach is supported by W3C and some researcher groups through the effort of SAWSDL[10]. The latter approach (the top-down approach) utilizes existing Web service technology as foundational systems, layering the semantics support on top of it. This approach is taken by several groups in academia and most distinguishing representatives are Web Services Modeling Ontology (WSMO)[11] [4] and OWL-S[12].

The Web Service Execution Environment (WSMX), presented in Figure 2, is an execution environment, which intends to realize the SWS lifecycle. It is a platform characterized by strong component decoupling, goal-driven Web service usage and direct support for mediation facilities. WSMX is a reference implementation of the WSMO that acts as the comprehensive conceptual model which describes various aspects of SWS consisting of formal descriptions of ontologies, web services, goals and mediators.

The COIN platform, as an extension of WSMX, is embracing WSMO and related set of languages, specifications and software environments.

2.2 Agents

Partners in inter-organizational collaborations are autonomous, socially cooperating and coordinating by exchanging information (sending messages) and share a need to adapt to changing environments. Thus, they display features which are often attributed to agents.

According to the definition of *weak agency* [10], the key properties of agents are:

- autonomy: agents are clearly identifiable problem solving entities with well-defined boundaries and interfaces which have control both over their internal state and over their own behaviour.
- reactivity: agents are situated (embedded) in a particular environment, i.e. they receive inputs related to the state of their environment through sensors. They then respond in a timely fashion, and act on the environment through effectors to satisfy their design objectives.

[10] http://www.w3.org/TR/sawsdl
[11] http://www.w3.org/Submission/WSMO
[12] http://www.w3.org/Submission/OWL-S

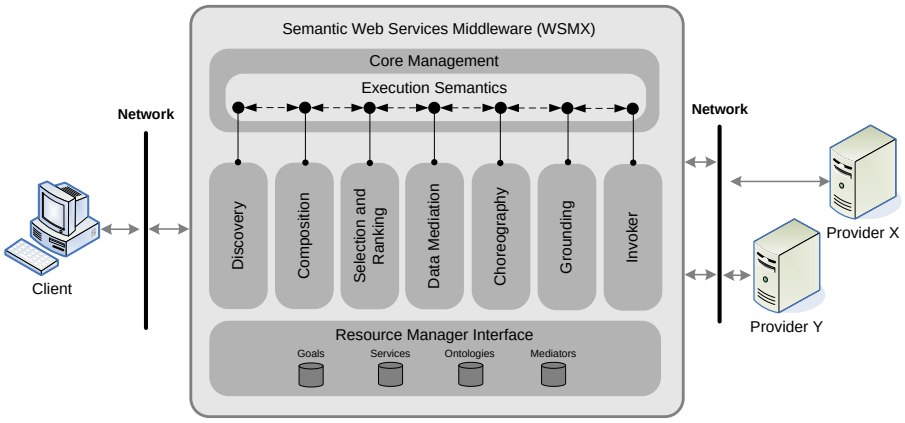

Fig. 2. Web Service Execution Environment

– pro-activeness: agents do not simply act in response to their environment, they are designed to fulfill a specific purpose, i.e. they have particular objectives (goals) to achieve. Agents are therefore able to exhibit goal-directed behaviour by taking the initiative and opportunistically adopting new goals.
– social ability: agents are able to cooperate with humans and other agents in order to achieve their design objectives.

Drawing these points together, the essential concepts of agent-based computing are: agents are capable of highly autonomous behaviour, representing encapsulations of computational entities and functions, high level inter-actions and organizational relationships within a society of agents situated in their environment.

2.3 Combining WSMX and Agents for Service Composition

WSMX as core component of the COIN platform offers functionality for discovery and execution of services which are available and registered in the platform. However, composition of services is in the responsibility of the user who has to make sure that the result of an atomic service invocation can be used e.g. for invoking another service. The idea of the COIN platform is to apply agents for service composition within the platform. Here, we can distinguish two aspects:

– Static composition at design time: The sequence and combination of required services is already known at design time. In this case, the service composition can be predefined and made accessible through the platform.
– Dynamic composition at run time: Services are discovered and composed at run time depending on a description of a complex goal.

Before we describe the details of the integration, in the following section present the scenario which we use for illustration.

3 Scenario

The high-level scenario is assuming the existence of an aircraft building company (e.g. Company X) which needs to assemble an EFIS[13] used to monitor and control flight signals and parameters. A typical EFIS installation consists of a number of devices such as data processors, flight display systems, data buses, and sensors. In order to accomplish the task Company X needs to purchase required components and arrange their transportation to its facilities. Additionally, Company X has specific preferences regarding the purchase and shipping tasks such as minimizing the overall price and/or time needed to receive the parts. The scenario further assumes presence of different selling and shipping companies capable of (partially) fulfilling Company X overall goal, as presented in Figure 3. It is worth noting that all the selling and shipping partners are exposing their business capabilities through Web service endpoints.

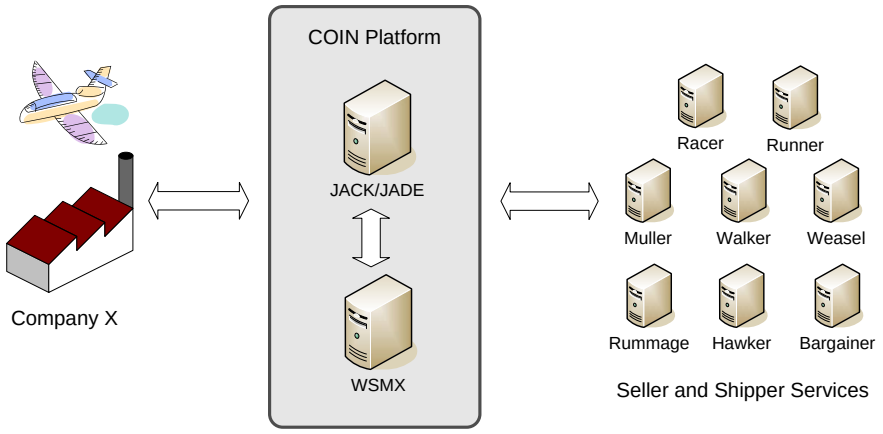

Fig. 3. Purchasing and shipping scenario

Taking into account a more concrete example Company X dispatches a goal to the platform as presented by the WSML code snippet shown in Listing 1.1. The goal formalizes the company's purchase (lines 16-24) and shipment (lines 25-26) preferences with some additional constraints (e.g. flight display price cannot exceed boundary of 7000 EUR as presented in line 20). The platform has a registry which enumerates a number of seller (namely Rummage, Bargainer and Hawker) and shipper (namely Runner, Walker, Racer, Muller and Weasel) services as presented in Figure 3. None of the services is capable of fulfilling the complete goal which makes cooperation between the SESA and the agent platform inevitable.

From this brief scenario description a number of properties that the platform needs to address can be derived:

[13] abbr. Electronic Flight Instrument System

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-rule"

namespace {_"http://www.coin-ip.eu/EFIS/ComplexGoal#",
  sop _"http://www.wsmo.org/sws-challenge/ShipmentOntologyProcess#",
  po _"http://www.coin-ip.eu/EFIS#"}

goal ComplexGoal

  importsOntology {
    sop#ShipmentOntologyProcess,
    po#ProductOntology}

  capability GoalA1Capability
    postcondition
      definedBy
( ?x[po#resX hasValue ?resX, po#resY hasValue ?resY,
  po#price hasValue ?xPrice] memberOf po#FlightDisplay
  and ?resX >= 1280
  and ?resY >= 768
  and ?xPrice <= 7000
  and ?y[po#processorMIPS hasValue ?MIPS,
  po#price hasValue ?yPrice] memberOf po#ProgrammableDisplayGenerator
  and  ?yPrice <= 9400
  and  ?MIPS >= 16 )
  and ?z[sop#price hasValue ?zPrice] memberOf sop#PriceQuoteResp
  and ?w[sop#price hasValue ?wPrice] memberOf sop#PriceQuoteResp
  and ?price=(?xPrice + ?yPrice+ ?wPrice + ?zPrice).
```

Listing 1.1. A snippet of the complex goal expressed in WSML

- *Goal-based discovery and invocation of services* - Company X needs to describe its goal in a formal way, independently of services. The platform should solve the goal by leveraging logical reasoning over descriptions of goals and services.
- *Service decomposition* - Company X describes a complex goal which has to be decomposed by the platform in order to find matching services.
- *Late service binding* - Company X doesn't know a priori which of the services (i.e. companies) will fulfill its goal. The platform should be capable to discover and invoke Web services at run-time according the the Company X's preferences (expressed through the goal description).
- *Data interoperability* - Company X and potential service candidates can express their requirements and capabilities in terms of different vocabularies and languages. The platform should provide support for data heterogeneities reconciliation.

4 Agent-based approach for service composition

The COIN platform provides the same interface and functionality as WSMX as core component. Services can be registered, made accessible (by mappings, e.g. lifting and lowering schema mappings), discovered and finally invoked by a user through the platform by sending concrete data to the service. A user sends his request to the platform in the form of a WSML goal which contains the data necessary to eventually fulfill his aims.

However, services registered in the platform are in general atomic services and are not automatically composed if a goal does not match a single service description. This limits the capability of the platform since in many cases *goals* as submitted by a user cannot be immediately satisfied by one service alone. These goals are complex, i.e. no single web service registered in the platform provides a sufficient functionality for fulfillment.

The approach we take for solving this restriction is based on the idea of goal decomposition. Goals are domain-specific, i.e. the ontological expressions used in the essential parts of goal description are based on domain ontologies. To specify a generic way of handling these domain-specific expressions is hardly realistic and feasible only for very simple scenarios. Therefore, we started with a more modest approach and gradually made it more generic.

The first step in our solution is to pre-define a goal decomposition for a specific domain. The assumption is that a complex goal as specified by a user can be syntactically split up into expressions defining subgoals which are not complex goals. The goal decomposition step takes a complex goal and generates a list of subgoals to be sent to the platform.

The second step consists of specifying the sequence in whioh the subgoals have to be processed. In a simple case, this sequence can be linear, but in more complex cases, the process could involve even nested interactions. A further complication in service composition is that inputs and outputs of service invocations are dependent on each other. A service B may require as input an output value of service A which has to be invoked before service B.

The sequence for subgoal processing is also predefined at design-time. The overall approach is depicted in Figure 4.

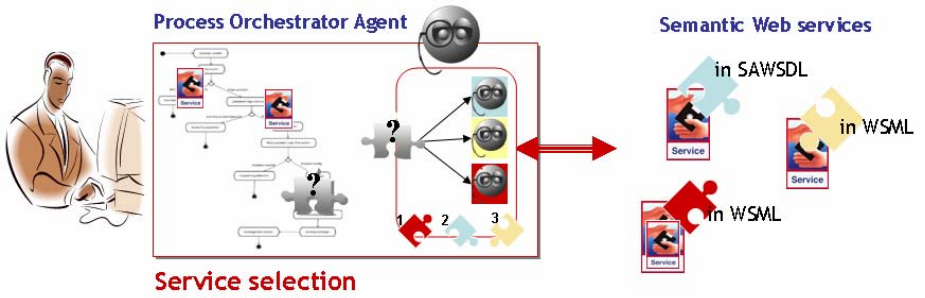

Fig. 4. Agents orchestrating Semantic Web services

In the following, we describe the approach with reference to the agent platform *JACK* [1], and then, using a model-driven approach, on a platform independent level.

4.1 Service composition with JACK

BDI agents [5] have a built-in theory of a planning from second principles approach to problems solving. BDI agents seem to be an ideal means for flexible service composition. On the one hand, BDI reasoning offers to the agents a flexible way to use the knowledge that is specified in the plan library of an agent. On the other hand, BDI plans can be expressed with a modelling language which nicely fits into a model driven approach to the specification of how service composition should be achieved. Flexibility w.r.t. service composition depends on the ability to seamlessly invoke services discovered at run-time. Conceptually, a BDI agent platform is very much in line with the WSML terminology, where a *Goal* specifies a service request. A BDI agent which sends a goal to WSMX is externalising the process of goal fulfillment instead of trying to achieve the goal on its own.

We decided to use the JACK Intelligent Agents development environment (JDE) for designing agents in the context of a service-oriented architecture. A major advantage of JDE is that it allows using graphical models for the specification of agent behaviours. We consider these models as the platform-specific level for agent design. Important to note is that these platform-specific models are directly executable in the runtime environment of JDE. BDI agents support adaptive execution which is introduced by flexible plan choice, in which the current situation in the environment is taken into account. A BDI agent has the ability to react flexibly to failure in plan execution, where both features are directly built into the framework of BDI reasoning.

For the scenario, the roles and teams presenting the actors are modeled first (see Figure 5). These include a $User$ (which in this case is WSMX, in fact), an $AgentComposition$ team for handling the (de-)composition, and teams and roles for both $Buyer$ and $Shipper$. The $User$ sends the complex goal (which includes concrete instance data needed for service invocation. e.g. an address for delivery) to the Composition team which decomposes the goal and sends a subgoal for buying to the $BuyerTeam$ and a shipment subgoal to the $ShipmentTeam$. Buyer and ShipmentTeam invoke the WSMX platform which handles the details of service discovery and invocation.

The domain-specific composition for the scenario is published as Web service and deployed into the COIN platform. The agent-based composition service provides an operation *achieveGoal* to WSMX which is invoked whenever no single service can satisfy a goal sent by a user of the COIN platform.

4.2 Service composition with PIM4Agents

In the service composition modeled with JACK, the agent model and (rather low-level) code related to service invocations is described on the same level. However, in order to separate levels of concern, an abstraction step towards independence of platforms-specific details is of benefit. Following the idea of model-driven development, a platform-independent level provides an abstraction

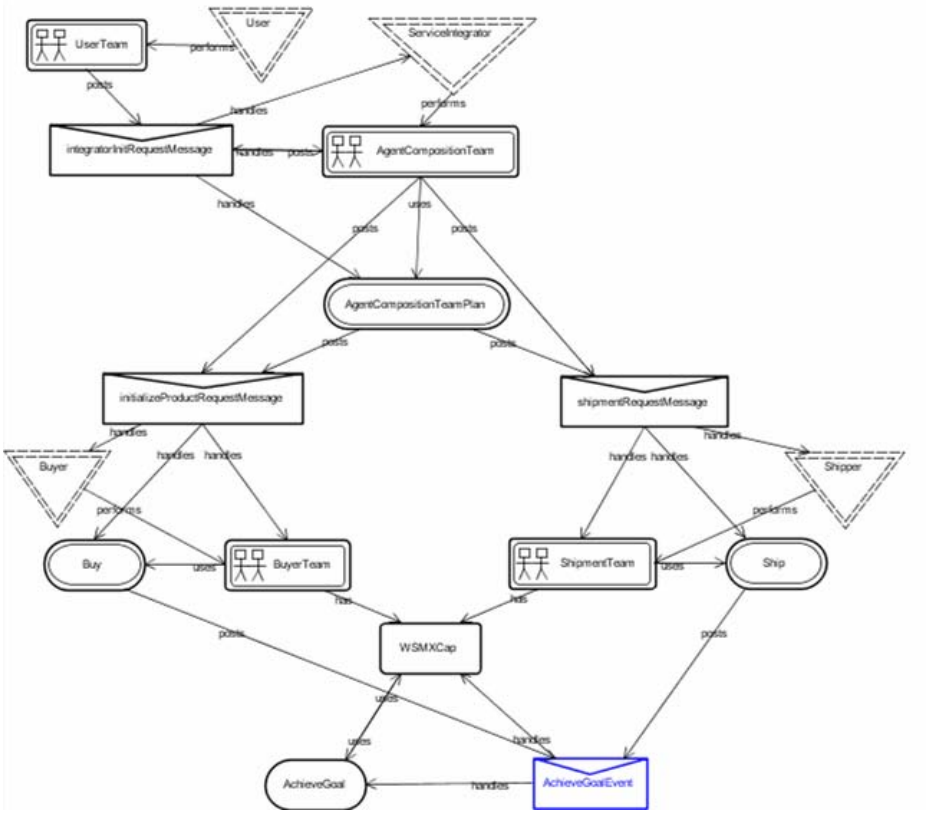

Fig. 5. Scenario entities modelled in JACK

from a concrete execution platform and details which pertain to a programming language (i.e. Java for JACK and Jade).

We use the *PIM4Agents* and its model editor [9] for describing the scenario on a platform-independent level. The model can then be transformed automatically to agent platforms (Jade and JACK are currently supported).

Again, the actors in the scenario are introduced first (see Figure 6). On the PIM level, we decompose the goal and sequentially invoke the WSMX platform. Hence, we distinguish between a $User$, a $CompositionAgent$ and the agent responsible for invoking the WSMX platform ($WSMXRequester$).

The $CompositionAgent$ has two plans at its disposal, one for decomposing the complex goal and one for handling the sequence of interactions when sending the subgoals to WSMX.

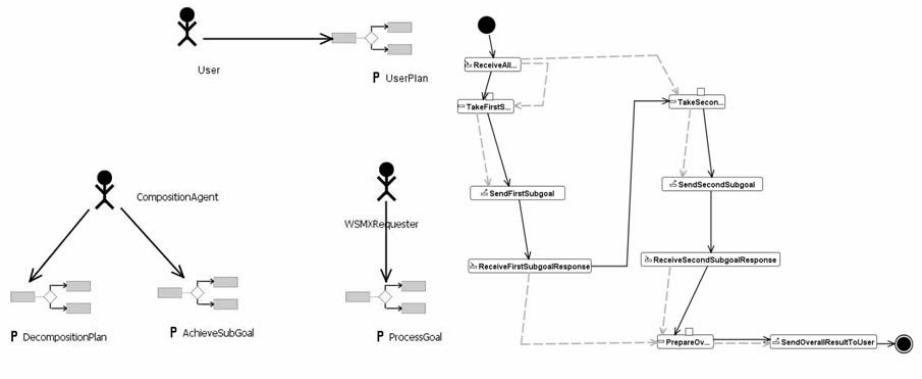

Fig. 6. Scenario modeled with PIM4Agents

5 Discussion and Open Issues

Agent-based service composition as presented in this paper follows the assumption that in business scenarios most parts of a workflow are fixed and can be modeled at design time. Service compositions have to be able to replace failing or inefficient services with different ones at runtime. When services inside a composition are dynamically discovered, selected and bound, it may arise a need to invoke services having an interface or protocol different from those originally expected by the service requested and to solve the mismatches at runtime. The integration with WSMX adds flexibility in the sense that (i) concrete interaction partners in a workflow step are only determined at runtime (late binding) and (ii) the details of the service invocation are handled by WSMX and hidden to the agent platform.

The approach presented in this paper is still centered around design time. In order to improve runtime flexibility, complex goals should be decomposed at runtime, depending on available services.

Currently, the Composition service is published as WSDL service. More consistent with the overall approach is to describe the domain-specific composition as Semantic Web service and register it in the platform.

6 Conclusion and Future Work

In this paper, we presented an approach to integrate an implementation of a SESA with agent platforms. The benefit of the integration is mutual since the Agent platform can leverage SESA when it comes to service discovery and invocation (i.e. late binding) and SESA can delegate complex goal fulfillment based on predefined goal decompositions to the agents. The integration approach is motivated and described in the context a high-level scenario coming from the aeronautical industry.

Dynamic decomposition of goals will be further investigated, in order to increase the adaptiveness of service compositions to changing environments.

Acknowledgments

The work published in this paper is (partly) funded by the E.C. through the COIN IP. It does not represent the view of E.C. or the COIN consortium, and authors are solely responsible for the paper's content. The authors wish to acknowledge the Commission for their support.

References

1. JACK Intelligent Agents, The Agent Oriented Software Group (AOS). http://www.agent-software.com/, 2006.
2. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, May 2001.
3. D. Fensel, M. Kerrigan, and M. Zaremba, editors. *Implementing Semantic Web Services: The SESA Framework*. Springer-Verlag, 2008.
4. D. Fensel, H. Lausen, A. Polleres, J. de Bruijn, M. Stollberg, D. Roman, and J. Domingue, editors. *Enabling Semantic Web Services: The Web Service Modeling Ontology*. Springer-Verlag, Berlin, 2007.
5. A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1991.
6. M. Stollberg, C. Feier, D. Roman, and D. Fensel. *Language Technology, Ontologies, and the Semantic Web*, chapter Semantic Web Services Concepts and Technology. Kluwer Publishers, 2006.
7. M. Turner, D. Budgen, and P. Brereton. Turning software into a service. *Computer*, 36(10):38–44, Oct. 2003.
8. T. Vitvar, M. Zaremba, M. Moran, M. Zaremba, and D. Fensel. SESA: Emerging Technology for Service-Centric Environments. *IEEE Software*, 24(6):56–67, November 2007.
9. S. Warwas and C. Hahn. The concrete syntax of the platform independent modeling language for multiagent systems. In *Proceedings of the International Workshop on Agent-based Technologies and applications for enterprise interoperability (ATOP 2008), Estoril, Portugal*, 2008.
10. M. J. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.