WAT 2009

PROCEEDINGS

of

CAEPIA 2009 Workshop on Agreement Technologies
(WAT 2009)

November 9th, 2009

Escuela Técnica Superior de Ingeniería Informática
- Sevilla, Spain

Marc Esteva, Alberto Fernández and Adriana Giret (Eds.)

www.agreement-technologies.org

Preface

Agreement is one of the crucial social concepts that helps human agents to cope with their social environment and is present in all human interactions. In fact, without agreement there is no cooperation and ultimately social systems cannot emerge. Agreement is necessary in our everyday life.

Until recently, the concept of agreement was a domain of study mainly for philosophers, sociologists and was only applicable to human societies. However, this situation has changed in the recent years, especially with the spectacular emergence of information society technologies. Computer science has moved from the paradigm of an isolated machine to the paradigm of a network of systems and of distributed computing. Likewise, artificial intelligence is quickly moving from the paradigm of an isolated and non-situated intelligence to the paradigm of situated, social and collective intelligence. Hence, the concept of agreement has become key for a robust understanding and an efficient implementation of artificial social systems.

In this context, Agreement Technologies is a new approach of Distributed Artificial Intelligence for constructing large-scale open distributed computer systems. This workshop on Agreement Technologies is specifically addressed for any work that aims at developing models, frameworks, methods and algorithms for constructing such systems. In other words, this workshop focuses on approaches and solutions for the needs of next generation computing systems where autonomy, interaction and mobility will be the key issues. Most importantly, it concentrates on techniques that enable software components to reach agreements on the mutual performance of services.

Agreement Technologies integrates many research efforts from different fields of Artificial Intelligence. Hence, this workshop is specifically tailored to research works related to this new approach. As CAEPIA is the leading Artificial Intelligence conference in Spain, it is a good forum to celebrate this workshop. Finally, the editors would like to thank all the people that bring about WAT and CAEPIA 2009. First of all, thanks to the authors for ensuring the richness of the workshop and the members of the program committee for their professionalism and dedication. Furthermore, we owe particular gratitude to the CAEPIA organizing committee.

Marc Esteva, Alberto Fernandez and Adriana Giret
WAT2009 Organizing Committee

ORGANIZING COMMITTEE

Marc Esteva (Artificial Intelligence Research Institute (IIIA), Spain)
Adriana Giret (Universidad Politécnica de Valencia, Spain)
Alberto Fernández (Universidad Rey Juan Carlos, Spain)

STEERING COMMITTEE

Carles Sierra (IIIA-CSIC, Spain)
Sascha Ossowski (Universidad Rey Juan Carlos, Spain)
Vicente Botti (Universidad Politecnica de Valencia, Spain)

PROGRAM COMMITTEE

Axel Polleres (DERI Galway, Ireland)
Adriana Giret (Universidad Politecnica de Valencia, Spain)
Alberto Fernandez (Universidad Rey Juan Carlos, Spain)
Carlos Ángel Iglesias (Universidad Politécnica de Madrid, Spain)
Carles Sierra (IIIA-CSIC, Spain)
David Pearce (Universidad Politécnica de Madrid, Spain)
Eugénio de Oliveira (Universidade do Porto, Portugal)
Eva Onaindía (Universidad Politecnica de Valencia, Spain)
Frank Dignum (Universiteit Utrecht, The Netherlands)
Helder Coelho (University of Lisbon, Portugal)
Holger Billhardt (Universidad Rey Juan Carlos, Spain)
Juan Antonio Rodriguez-Aguilar (IIIA-CSIC, Spain)
Juan Manuel Serrano (Universidad Rey Juan Carlos, Spain)
Jaelson Castro (Universidade Federal de Pernambuco, Brazil)
Luis Botelho (ISCTE, Lisbon, Portugal)
Maite Lopez-Sanchez (Universitat de Barcelona, Spain)
Marc Esteva (IIIA-CSIC, Spain)
Maria Jose Ramirez (Universidad Politecnica de Valencia, Spain)
Michael Wooldridge (University of Liverpool, UK)
Pablo Noriega (IIIA-CSIC, Spain)
Sascha Ossowski (Universidad Rey Juan Carlos, Spain)
Vicente Julián (Universidad Politecnica de Valencia, Spain)
Vicente Botti (Universidad Politecnica de Valencia, Spain)
Viviane Torres da Silva (Universidade Federal Flumiense, Brazil)
Wamberto Vasconcelos (University of Aberdeen, UK)

Index

Negotiation with Price-dependent Probability Models *

Antonio Bella, Cèsar Ferri, José Hernández-Orallo, and María José Ramírez-Quintana

Universidad Politécnica de Valencia, DSIC, Valencia, Spain

Abstract. Negotiation and agreement generally require models of the peers who are involved in the negotiation. One typical area where negotiation takes place is in selling and retailing, which is also known as Customer Relationship Management (CRM). Customers and products are usually modelled using previous retailing experiences with similar or dissimilar customers and products. Machine learning is typically used to construct these models, which can be used to design mailing campaigns, to recommend new products, to do cross-selling, etc. Many CRM problems can already be solved through rankers, recommender systems, etc., provided that there are good models of customer and product behaviours available. A related but more general problem is when models are used to negotiate with one or more features of the product (or, less frequently, the customer) such as prices, bonuses, warranties, etc. Additionally, if it is possible to make several bids until an agreement is reached, methods must be devised so that the maximum profit is obtained by the seller. In this work, we present a taxonomy of CRM problems, from which we distinguish those that have already been solved and those whose solutions are still pending. Then, we extend classical purchase probability rankings to the notion of profit probability curves (price-dependent distributions), and we propose a simple negotiation solution for these cases.

Keywords: negotiation, agreement, bargaining, CRM, ranking, probability estimation, negotiable features, machine learning.

1 Introduction

The evolution of small-sized retailing to mass Customer Relationship Management (CRM) has usually implied greater automation of the selling process, the selection of products, customer prescriptions, cross-selling, up-selling, market segmentation, etc. Techniques such as mailing campaign design, recommender systems, and others (e.g. data-mining and machine learning [5] from the area of business intelligence) are nowadays common in these applications. However, this evolution has not usually included traditional techniques in selling such as

* This work has been partially supported by the EU (FEDER) and the Spanish MEC/MICINN, under grant TIN 2007-68093-C02 and the Spanish project "Agreement Technologies" (Consolider Ingenio CSD2007-00022).

bargaining and other kinds of negotiation because of the difficulty of automating them and also because more general models about customer behaviour are needed. With the diversification and specialisation of services, there is a need for new techniques that deal with selling scenarios where price (or any other negotiable feature) can be adjusted to the customer (price discrimination using custom discounts and offers) in order to achieve the goal of mass customisation.

Consider, for instance, a classical mailing campaign design or market targeting problem. Given n equal products and m customers, the subset of customers to whom a product must be offered (customer prescription) must be devised. In order to do this, machine learning is usually employed to learn probabilisitc models, which assess the probability of buying for each customer, from which a ranking of customers is performed. If the campaign has both fixed and variable costs, there are simple techniques (e.g. profit lifts) to calculate the subset of customers that maximises the expected profit [4]. Similar techniques can be used if there are many kinds of products where several rankings and constraints (stocks) must be taken into account [2].

A very different type of problem, however, is when product or customer buying expectations can be influenced by changing one or more features of the product or the customer, such as prices, bonuses, warranties, etc. For instance, the probability of a customer buying a product changes dramatically if the price is modified. And it is frequent to adjust prices (or, in a subtler way, to offer discounts) to those customers who are less eager to buy a product. Setting different prices for various customers can allow us to maximise profit.

In this scenario, we need to move from probabilistic models (where, given a product and a customer, we have a probability of buying) to profit probability curves or price-dependent distributions (where, given a product and customer, we see the evolution of the probability of buying according to a certain feature (e.g. price)). This more complex scenario has not been addressed to date.

Additionally, negotiating with price or other negotiable features normally implies that counter-offers from both buyer and seller are allowed up to a maximum number of bids or until an agreement is reached. In other words an offer can be made to a customer at a given price, and if the customer does not buy the product, a lower price (a special discount offer for that particular customer) can be offered. This is the beginning of a negotiation process between the seller and the buyer that we have already studied in [3]. In that work, we developed several methods to derive the profit probability curves, and we also introduced new negotiation strategies.

However, in general, the case when a negotiation can take place, in parallel, for several products and customers must also be solved. In this case, not only must the profit probability curves for each pair of product and customer be derived, but is also necessary to analyse how to combine these curves to develop new optimal negotiation strategies in these much more complex scenarios.

This work attempts to address this problem. We place ourselves on the side of the seller. Since the main overall objective is to sell as much as possible with the highest possible net price. Therefore, agreements are considered to be optimal

when they are optimal for the seller. Nonetheless, the techniques that we discuss in this paper can also be used when both buyer and seller model each other, or when a buyer wants to negotiate with several sellers.

The paper is organised as follows. In Section 2, we devise a taxonomy of CRM prescription problems, from which we distinguish those that have been solved in the past and those which we propose a solution. In Section 3, we present a detailed description of a specific scenario with one kind of product, a negotiable price, and M customers, and explain how it is possible to solve other kinds of CRM prescription problems using the same strategy. In Section 4, we present our conclusions and future work.

2 A Taxonomy of CRM Prescription Problems

There are a great variety of different CRM prescription problems that can be defined only by taking into account the cardinality of the different kinds of products and customers ($1{-}1, N{-}1, 1{-}M, N{-}M$) and the presence or absence of negotiation. As we have stated in the introduction, if a feature is negotiable, then we can introduce some kind of negotiation into the CRM process; however, if it is non-negotiable (fixed), then we are dealing with a traditional CRM prescription problem. Focusing on the seller, in this paper, when we refer to the price of a product we mean the net price that the seller obtains for the product, e.g., if a customer buys a product for 3 euros, the net price is 3 euros since fixed and variable costs are not included (except if mailing is involved).

In any of these situations, data-mining techniques can help the seller by modelling customer behaviour in order to make good decisions. In this context, that means obtaining as much profit as possible.

Table 1. Different CRM prescription problems that consider the number of different kinds of products to sell, whether the net price for the product is fixed or negotiable, and the number of customers.

Case	Kinds of products	Net price	Number of customers	Approach
1	1	fixed	1	Trivial
2	1	fixed	M	Customer ranking [4]
3	N	fixed	1	Product ranking [4]
4	N	fixed	M	Joint Cut-off [2]
5	1	negotiable	1	Negotiable Features [3]
6	1	negotiable	M	This work
7	N	negotiable	1	This work
8	N	negotiable	M	Future work

Table 1 shows eight different CRM prescription problems that are defined by considering the number of products and customers involved as well as the fixed or negotiable nature of the net price for each product. The last column shows several approaches, that have already been proposed in the literature for solving some of these problems. It also shows others that are proposed in this paper for solving the remainder. We discuss each of them in more detail below.

2.1 CRM Prescription Problems without Negotiation

Case 1 in Table 1 (one kind of product, fixed net price, and one customer) is trivial. In this scenario, the seller offers the product to the customer at a fixed price and the customer may or not buy the product. The seller cannot do anything more because s/he does not have more products to sell. S/he cannot negotiate the price of the product with the customer, and s/he does not have any more customers for the product.

Case 2 in Table 1 (one kind of product, fixed net price, and M customers) is the typical case of a mailing campaign design. The objective is to obtain a customer ranking to determine the set of customers to whom the mailing campaign should be directed in order to obtain the maximum profit. Data-mining can help in this situation by learning a probabilistic estimation model from previous customer data that includes information about similar products that have been sold to them. This model will obtain the buying probability for each customer, so by putting them in order of decreasing buying probability, the most desirable customers will be at the top of the ranking. Using a simple formula for marketing costs, we can establish a threshold/cut-off in this ranking. The customers above the threshold will be offered the product. This is usually plotted using the so-called lift charts.

Case 3 in Table 1 (N kind of products, fixed net price, and one customer) is symmetric to case 2. Instead of N customers and one product, in this case, there are N different products and only one customer. The objective is to obtain a product ranking for the customer. Similarly, data-mining can help to learn a probabilistic estimation model from previous product data that have been sold to similar customers. This model will predict the buying probability for each product, so by putting them in order of decreasing buying probability, the most desirable products for the customer will be at the top of the ranking. This case overlaps to a great extent with recommender systems.

Case 4 in Table 1 (N kinds of products, fixed net price, and M customers) is studied in [2]. This situation is more complex than the cases 2 and 3, since there is a data-mining model for each product. In other words, there are N rankings of customers (one for each product) and the objective is to obtain the set of customers that gives the maximum overall profit. Note that, normally, the best local cut-off of each model (the set of customers that gives the maximum profit for one product) does not give the best global result. Moreover, several constraints would have to be fulfilled (limited stock of products, the customers can only buy one product), which usually happens in real situations. Two different methods are proposed in [2] to obtain the global cut-off: one is based on merging the prospective customer lists and using the local cut-offs, and the other is based on simulation. The study in [2] shows that use simulation to adjust model cut-off obtain better results than more classical analytical methods.

2.2 CRM Prescription Problems with Negotiation

In the above four cases, we have assumed that the net price of the product is fixed. When this is not the case, the objective of the sellers changes. They do not

have to sell the product, but they have to sell it at the maximum price, which the seller and the buyer can negotiate. Figure 1 (Left) shows how the behaviour of a customer can be approximated. The customer buys the product if the price is less than or equal to the maximum price that s/he could pay for it. Figure 1 (Right) shows the expected profit for each price (the expected profit is equal to the buying probability multiplied by the price). In this case, it is clear that the maximum expected profit is obtained when the seller sells the product at the maximum price that the customer can pay.

We will show later, that the seller does not know this real model and tries to learn the most accurate model from previous data by using data-mining techniques.

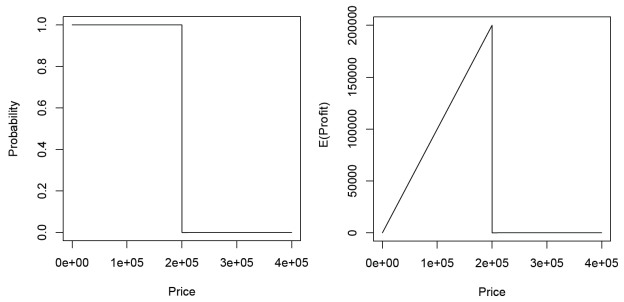


Fig. 1. Real behaviour of the buying model of a customer. **Left**: Probability distribution function. **Right**: Associated expected profit.

In [3], a formal definition of a negotiable feature is given and a case with one kind of product, a negotiable price, and one customer is studied (Table 1, case 5). In this case, there is one probabilistic data-mining model for the customer, which was learned from previous product data. Several data-mining algorithms and several negotiation strategies were studied experimentally. As can be observed in [3] the best results were obtained by approaching the buying model curve by the cumulative distribution function of a normal distribution. The mean μ is equal to the value obtained by a regression model and the standard deviation σ is equal to the standard error of the model (mean absolute error, *mae*). Figure 2 shows an example of a probabilistic buying model of a customer who is interested in buying a flat that is approximated by a normal distribution with $\mu = 200,000$ and $\sigma = 20,000$. As can be observed in this figure, the mean price has a probability of 0.5, so the maximum expected profit is located just to the left of that point (the seller will decrease the price to increase the probability of buying the product if s/he can only make one offer to the customer). If the seller can make several offers, then the offers can be distributed along the curve to maximise the profit of the product, according to several negotiation strategies.

There are some details that should be taken into account from our work in [3]. Note that the normal distribution is limited on the left, when working with the expected profit curve (i.e., the probability density function) because the expected profit is zero when the price of the product is zero. Also note that, in [3], only symmetric normal distributions were considered, but it would also be interesting

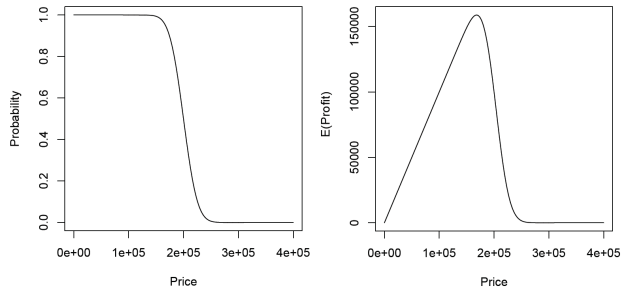


Fig. 2. Probabilistic buying model of a customer approximated by a normal distribution with $\mu = 200,000$ and $\sigma = 20,000$. **Left**: Probability distribution function. **Right**: Associated expected profit.

to study non-symmetric normal distributions (i.e., to consider different standard deviations on the left and on the right of the mean, as is common in real life. For example, it is easy to buy a product simply because it is cheap (although not essential), therefore, in this situation the standard deviation on the left will be greater than the one on the right. An example of a skew-normal distribution [1] can be seen in Figure 3.

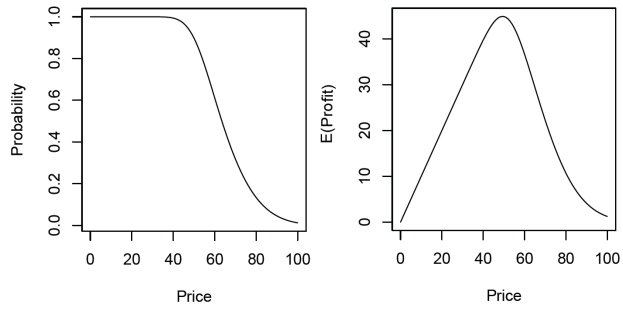


Fig. 3. Probabilistic buying model of a customer approximated by a skew-normal distribution with location $\xi = 50$, scale $\omega = 2$, and shape $\alpha = 3$. **Left**: Probability distribution function. **Right**: Associated expected profit.

The cases 6, 7 and 8 in Table 1 have not yet been studied. However, they can be understood as an extension of case 5 combined with the rankings of customers and products that are used in the approaches proposed in Table 1 for the cases 2 and 3.

In case 6 in Table 1 (one kind of product, a negotiable price, and M customers), there is a curve for each customer, that is similar to the curve in case 5 (Figure 4, Left). If the seller can only make one offer to the customers, the seller will offer the product at the price that gives the maximum expected profit (in relation to all the expected profit curves) to the customer whose curve achieves the maximum. However, if the seller can make several offers, the seller will distribute the offers along the curves following a negotiation strategy. In this case, the seller not only changes the price of the product, but the seller can also change the customer that s/he is negotiating with, depending on the price of the product (that is, by selecting the customer in each bid who gives the greatest expected

profit at this price). Therefore, these curves can be seen as an evolution of the customer ranking for each price.

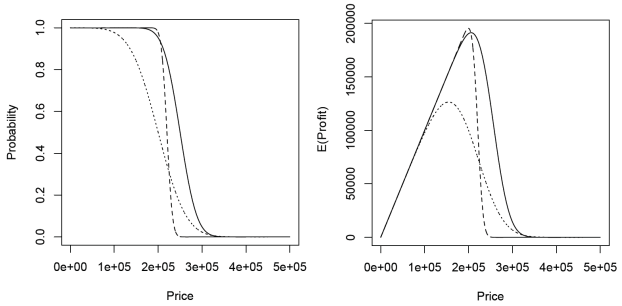


Fig. 4. Probabilistic buying models of 3 different customers approximated by 3 normal distributions with $\mu_1 = 250,000$ and $\sigma_1 = 30,000$, $\mu_2 = 220,000$ and $\sigma_2 = 10,000$, and $\mu_3 = 200,000$ and $\sigma_3 = 50,000$. **Left**: Probability distribution function. **Right**: Associated expected profit.

Case 7 in Table 1 (N kind of products, a negotiable price, and one customer) is symmetric to case 6. Instead of one curve for each customer, there is one curve for each product that was learned from the previous customer data. In this case, the curves represent a ranking of products for that customer. The learned data-mining models will help the seller to make the best decision about which product the seller offers to the customer and at what price. Figure 4 is an example of this case since the curves would represent three different products to be offered to one customer.

Case 8 in Table 1 (N kind of products, a negotiable price, and M customers) is the most complex of all. There is one data-mining model for each product and customer (i.e., $N \times M$ curves). The objective is to offer the products to the customer at the best price in order to obtain the maximum profit. Multiple scenarios can be proposed for this situation: each customer can buy only one product; each customer can buy several products; if the customer buys something, it will be more difficult to buy another product; there is limited stock; etc.

To solve cases 6, 7 and 8, we propose extending the classical concept of ranking customers or products to profit probability curves in order to obtain a ranking of customers or products for each price (similar to cases 2 and 3). For example, Figure 4 shows that, for a price of 300,000 euros the most desirable customer is the one represented by the solid line, the second most desirable one is the customer represented by the dotted line, and the least desirable customer is the one represented by the dashed line. The situation changes for a price of 200,000 euros; at that point the most desirable customer is the one represented by the dashed line, the second most desirable one is the customer represented by the solid line, and the least desirable customer is the one represented by the dotted line. Therefore, an important property of these probabilistic buying models is that there is a change in the ranking at the point where two curves cross.

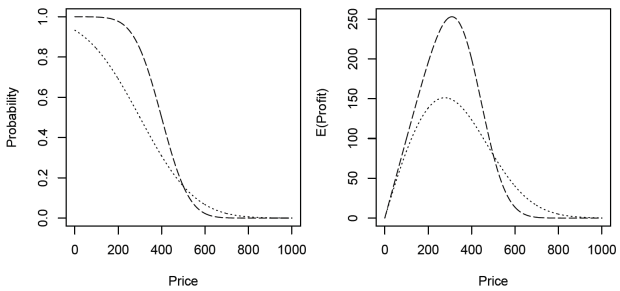


Fig. 5. Probabilistic buying models of 2 different customers approximated by 2 normal distributions with $\mu_1 = 400$ and $\sigma_1 = 100$, and $\mu_2 = 300$ and $\sigma_2 = 200$. **Left**: Probability distribution function. **Right**: Associated expected profit.

3 Scenario with Negotiable Price and several Customers

To study case 6 in more depth, we start with the simplest situation with two customers, and we explain the negotiation strategy that the seller will follow by means of an example.

In Figure 5, there are two curves representing the buying model of two different customers. The buying model of the first customer follows a normal distribution with $\mu_1 = 400$ and $\sigma_1 = 100$, and it is represented by a dashed line. The buying model of the second customer follows a normal distribution with $\mu_2 = 300$ and $\sigma_2 = 200$, and it is represented by a dotted line. These are the models; however, the real situation is that the maximum buying price for customer 1 is 100 euros and 150 euros for customer 2.

We assume a simple negotiation process for this example. The negotiation strategy that we describe is similar to the Best Local Expected Profit (BLEP) strategy explained in [3], but in that case the number of offers was limited to n.

Table 2. Left: Trace of the negotiation process. **Right:**Trace of the negotiation process with the ordering pre-process.

Offer	Price	Customer	Accepted
1	309	1	No
2	214	1	No
3	276	2	No
4	149	1	No
5	101	1	No
6	150	2	Yes

Offer	Price	Customer	Accepted
1	309	1	No
2	276	2	No
3	214	1	No
4	150	2	Yes

The strategy consists of offering the product at the price that obtains the maximum expected profit for the customer whose curve reaches the maximum. If the customer accepts the offer, the process is finished. If the customer does not accept the offer, his/her curve is normalised taking into account the following: the probabilities of buying that are less than or equal to the probability of buying at this price will be set to 0; and the probabilities greater than the probability of buying at this price will be normalised between 0 and 1. This process is

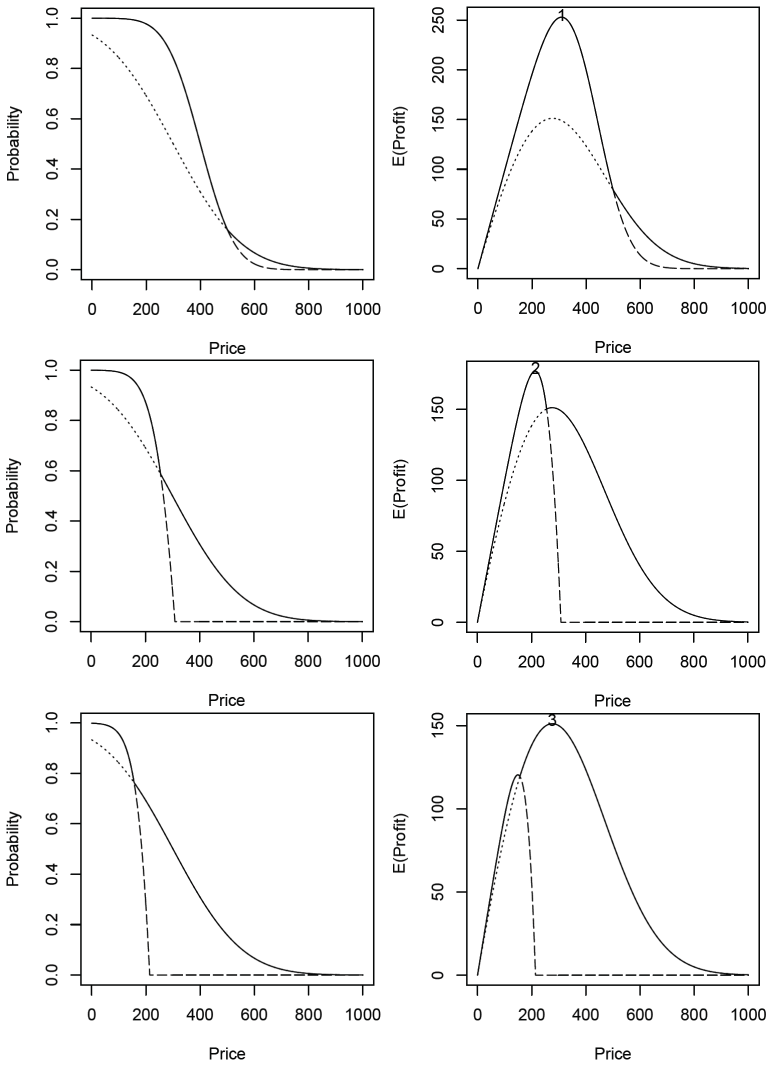


Fig. 6. Points 1, 2 and 3 in the negotiation process. **Left**: Probability distribution function. **Right**: Associated expected profit.

repeated with each customer until one of them accepts an offer[1]. The trace of the negotiation process is described in Table 2 (Left) and shown graphically in Figures 6 and 7. In each iteration, the maximum of the functions is calculated (the envelope curve). The envelope curve is represented by a solid line in Figures 6 and 7.

Note that as Table 2 (Left) shows, the third offer is greater than the second one. This is because there is more that one customer in the negotiation process and the offer is made at the price that maximises the expected profit at each iteration. Therefore, it is easy to propose an improvement for this negotiation strategy with a limited number of offers, which is similar to BLEP with n bids. This improvement is a pre-process that consists of calculating the n points and ordering them by the price before starting the negotiation. Following the example shown in Table 2 (Left), if there are only 4 bids no one will buy the product. However, with our improvement (the pre-process) customer 2 will buy the product at a price of 150 euros as shown in Table 2 (Right).

This negotiation scenario suggests that other negotiation strategies can be proposed for application to problems of this type in order to obtain the maximum profit. One problem with the BLEP strategy is that it is very conservative. It might be interesting to implement more aggressive strategies that make offers at higher prices (graphically, more to the right). A negotiation strategy that attempts to do this is one of the strategies proposed in [3], the Maximum Global Optimisation (MGO) strategy (with n bids). The objective of this strategy is to obtain the n offers that maximise the expected profit by generalising an optimisation formula that was presented in [3].

In case 6, we have presented an example with two customers and one product, but it would be the same for more than two customers. In the end, there would be one curve for each customer, and the same negotiation strategies could be applied.

Case 7 (N kind of products, a negotiable price, and one customer) is the same as case 6, but the curves represent the buying model of each product for each customer, and a ranking of products will be obtained for each price.

Case 8 (N kind of products, a negotiable price, and M customers) can be studied using the same concept of expected profit curves, but there will be $N \times M$ curves. The use of simulation or some kind of evolutionary computation will be necessary to obtain a good solution because case 8 is similar to case 4 where the best point in each curve does not give the best global solution. For each of the N kind of products, there will be M curves that belong to the buying model of each customer. Figure 8 presents a simple example with two products and two customers, where a customer can only buy a maximum of one product. In this example, customer 1 (dashed line) has the maximum expected profit for both products, corresponding to 80 euros for product 1 and 112 euros for product 2. The best local decision would be to offer product 2 to customer 1; however, customer 2 (dotted line) has the maximum expected profit of 27 euros

[1] More stop conditions are possible (e.g. limited number of offers (BLEP with n bids), minimum selling price, etc.)

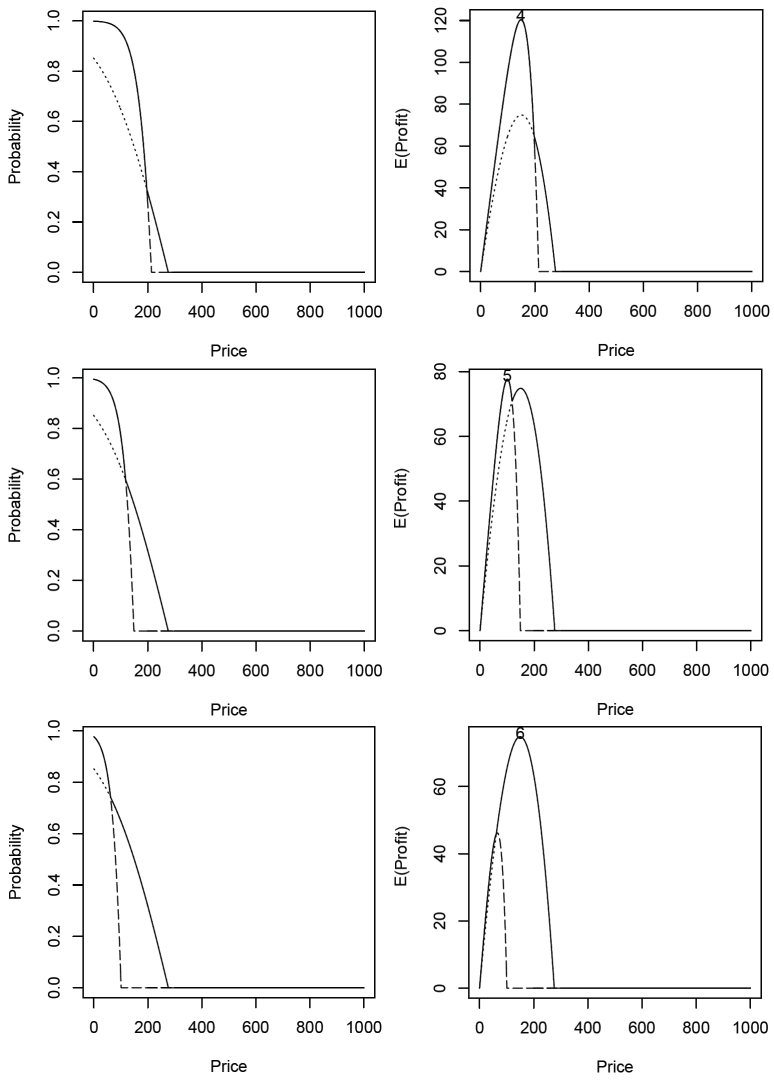


Fig. 7. Points 4, 5 and 6 in the negotiation process. **Left**: Probability distribution function. **Right**: Associated expected profit.

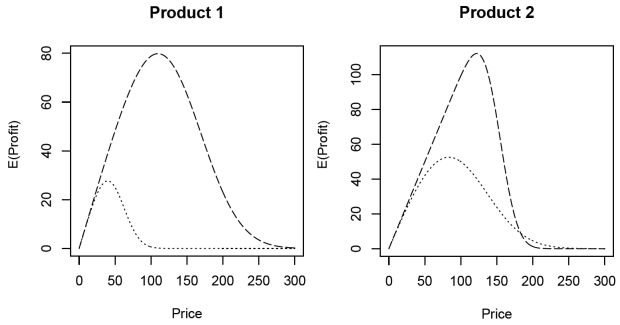


Fig. 8. Example of probabilistic models of two different customers. **Left**: Product 1. **Right**: Product 2.

for product 1 and 52 euros for product 2. Thus, for a better global solution, it would be better to offer product 2 to customer 2 and product 1 to the customer 1 since the overall profit would be greater.

4 Conclusions

In this paper, we have devised a taxonomy of CRM prescription problems, where automated learning can help a seller to make a good decision about which product should be offered to which customer and at what price in order to obtain as much overall profit as possible.

Some of these problems have already been studied, and we have explained the approaches proposed to solve them. In the cases that have not yet been studied (negotiable price, several products and/or several customers), we have proposed a solution based on the extension of rankings to expected profit curves, in which there is a ranking of customers and/or products for each price of the product.

As future work, we plan to study the performance of the proposed methods with experiments applying the negotiation strategies described in Section 3 and other suitable negotiation strategies to cases 6, 7 and 8 shown in Table 1.

References

1. A. Azzalini. A class of distributions which includes the normal ones. *Scandinavian Journal of Statistics*, 12:171–178, 1985.
2. A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. Joint cut-off probabilistic estimation using simulation: A mailing campaign application. In *IDEAL*, volume 4881 of *LNCS*, pages 609–619. Springer, 2007.
3. A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. Feature Dependent Models. Technical Report http://users.dsic.upv.es/∼abella/papers/FDM.pdf, Universidad Politécnica de Valencia, 2009.
4. M.J.A. Berry and G.S. Linoff. *Mastering Data Mining: The Art and Science of Customer Relationship Management.* Wiley, 1999.
5. T.M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

The THOMAS architecture: A case study in Home Care Scenarios.

Fraile Nieto, J.A.[1], Rodríguez, S. [2], Bajo, J.[1] and Corchado, J.M.[2]

[1]Pontifical University of Salamanca, c/ Compañía 5, 37002 Salamanca, Spain
[2]University of Salamanca, Plaza de la Merced s/n, 37008 Salamanca, Spain
{jafraileni, jbajope}@upsa.es, {srg, corchado}@usal.es

Abstract. Nowadays, the need for architectures and computational models for large scale open multi-agent systems is considered a key issue for the success of agent technology in real world scenarios. The main goal of this paper is to describe a case study in Home Care applying an abstract architecture and a computational model for large scale open multi-agent systems based on a service-oriented approach. The architecture we used is THOMAS. THOMAS is specifically addresses to design organizational structures for multiagent systems, in this case, a Home Care system. The paper presents services example for the management of a home dependent environment, which demonstrates the new features of the proposal.

Keywords: Dependent environments, Abstract Arquitectures, Multiagent Systems, Home Care, Organizations, Services

1 Introduction

The continuous growth of the dependency people sector has dramatically increased the need for new home care solutions [1] [7]. Besides, the commitments that have been acquired to meet the needs of this sector, suggest that it is necessary to modernize the current systems. Home Care is one of the objectives of the pervasive computing, and dependent people require new solutions that can make use of the technological advances to provide novel and fundamental services [2]. The vision of the pervasive computing allows improving the quality, access, equity and continuity of health care [2]. In this sense, the intelligent environments can improve health care services and can have a high social impact, especially in the home care services for dependent chronic patients [3]. Home Care requires effective communication as well as distributed problem solving [3].

Multi-agent systems [4], [15], and intelligent devices-based architectures have been recently explored as supervisor systems for health care scenarios [2] for elderly people and for Alzheimer patients [7]. These systems allow providing constant care in

the daily life of dependent patients [5], predicting potentially dangerous situations and facilitating a cognitive and physical support for the dependent patient [3].

The goal of work is to present a case study in which the THOMAS (MeTHods, Techniques and Tools for Open Multi-*A*gent Systems) [6] [9] architecture is used to build an open MAS for supervising and monitoring dependent patients at home. THOMAS is a new architecture for open MAS and is made up of a group of related modules that are well-suited for developing systems in volatile environments. THOMAS provides a high level of abstraction to determine which components are necessary for addressing all of the needs and characteristics of a home care environment. The multi-agent system developed offers a series of functionalities including automatic reasoning and planning mechanism for scheduling the medical staff working day, an alert system, a location and tracking system and an identification system. The medical staffs has been provided with PDAs and mobile phones, as well as with Java Card tags, and the home environments have been equipped with presence detection sensors, access control mechanisms, door opening devices and video cameras. The multi-agent system monitors the daily routine of the patient and detects dangerous situations. If any anomalous situation is detected, alert system is used to obtain medical assistance.

One of the objectives of MAS is to build systems capable of autonomous and flexible decision-making, and that will cooperate with other systems within a "society" . This "society" must consider characteristics such as distribution, continual evolution and flexibility, all of which allow the members (agents) of the society to enter and exit, to maintain a proper structural organization, and to be executed on different types of devices. All of these characteristics are incorporated in THOMAS via the open MAS and virtual organization paradigm, which was conceived as a solution for the management, coordination and control of agent performance. The organizations not only find the structural composition of agents (i.e., functions, relationships between roles) and their functional behaviour (i.e., agent tasks, plans or services), but they also describe the performance rules for the agents, the dynamic entrance and exit of components, and the dynamic formation of groups of agents.

The rest of the paper is structured as follows: section 2 provides an analysis of related studies; section 3 presents the proposed architecture model; section 4 shows an example of an implementation, highlighting the new possibilities provided by this type of architecture and specifically presents an approach for a home care management; finally, some conclusions of work are shown in section 5.

2 Related works

Agents and multi-agent systems in dependency environments are becoming a reality, especially in health care. Most agents-based applications are related to the use of this technology in the monitoring of patients, treatment supervision and data mining. Lanzola present a methodology [10] that facilitates the development of interoperable intelligent software agents for medical applications, and propose a generic computational model for implementing them. The model may be specialized in order to support all the different information and knowledge-related requirements of a

hospital information system. Meunier proposes [11] the use of virtual machines to support mobile software agents by using a functional programming paradigm. This virtual machine provides the application developer with a rich and robust platform upon which to develop distributed mobile agent applications, specifically when targeting distributed medical information and distributed image processing. While an interesting proposal, it is not viable due to the security reasons that affect mobile agents, and there is no defined alternative for locating patients or generating planning strategies. There are also agents-based systems that help patients to get the best possible treatment, and that remind the patient about follow-up tests [12]. They assist the patient in managing continuing ambulatory conditions (chronic problems). They also provide health-related information by allowing the patient to interact with the on-line health care information network. Decker & Li propose [8] a system to increase hospital efficiency by using global planning and scheduling techniques. They propose a multi-agent solution that uses the generalized partial global planning approach which preserves the existing human organization and authority structures, while providing better system-level performance (increased hospital unit throughput and decreased impatient length of stay time). To do this, they use resource constraint scheduling to extend the proposed planning method with a coordination mechanism that handles mutually exclusive resource relationships. Other applications focus on home scenarios to provide assistance to elderly and dependent persons. RoboCare presents a multi-agent approach that covers several research areas, such as intelligent agents, visualization tools, robotics, and data analysis techniques to support people with their daily life activities [13]. TeleCARE is another application that makes use of mobile agents and a generic platform in order to provide remote services and automate an entire home scenario for elderly people [4].

The architecture we used is THOMAS (MeTHods, techniques and tools for Open Multi-Agent Systems) [6] [9], which is composed of a set of related modules that are appropriate for developing systems in highly volatile environments similar to the one presented in this study. This paper presents the main characteristics of THOMAS as well as the results obtained after having applied the system to a case study..

3 THOMAS Architecture Model

THOMAS architecture basically consists of a set of modular services. Though THOMAS feeds initially on the FIPA[1] architecture, it expands its capabilities to deal with organizations, and to boost its services abilities. In this way, a new module in charge of managing organizations has been introduced into the architecture, along with a redefinition of the FIPA Directory Facilitator that is able to deal with services in a more elaborated way, following Service Oriented Architectures guidelines. As has been stated before, services are very important in this architecture. In fact, agents have access to the THOMAS infrastructure through a range of services included on different modules or components. The main components of THOMAS are the following [9]:

[1] http://www.fipa.org (Foundation for Intelligent Physical Agents)

- Service Facilitator (SF), this component offers simple and complex services to the active agents and organizations. Basically, its functionality is like a yellow page service and a service descriptor in charge of providing a green page service. The SF acts as a gateway to access the THOMAS platform. It manages this access transparently, by means of security techniques and access rights management. The SF can find services searching for a given service profile or searching for the goals that can be fulfilled when executing the service. This is done using the matchmaking [14] and service composition mechanisms [7] which are provided by the SF. The SF also acts as a yellow pages manager and in this way it can find which entities provide a given service.
- Organization Management System (OMS), mainly responsible for the management of the organizations and their entities. Thus, it allows the creation and management of any organization. The OMS is in charge of organization life-cycle management, including specification and administration of both their structural components (roles, units and norms) and their execution components (participant agents and roles they play, and active organizational units). Organizations are structured by means of organizational units, which represent groups of entities (agents or other units), which are related in order to pursue a common goal. These organizational units have an internal topology (i.e. hierarchical, team, plain), which imposes restrictions on agent relationships and control (ex. supervision or information relationships).
- Platform Kernel (PK), it maintains basic management services for an agent platform. The PK is in charge of providing the usual services required in a multi-agent platform. Therefore, it is responsible for managing the life-cycle of the agents included in the different organizations, and it also makes it possible to have a communication channel (incorporating several message transport mechanisms) to facilitate interaction among entities. On the other hand, the PK provides safe connectivity and the mechanisms necessary for allowing multi-device interconnectivity.

From a global perspective, the THOMAS architecture offers a total integration enabling agents to transparently offer and request services from other agents or entities, at the same time allowing external entities to interact with agents in the architecture by using the services provided.

4 Applying THOMAS to Home Care

The Home Care example is an application that facilitates the interconnection between dependent people and their environment and medical staff (doctors, nurses and personal assistant), delimiting services that each one can request or offer. The system controls which services must be provided by each agent. The internal functionality of these services is the responsibility of provider agents. However, the system imposes some restrictions regarding service profiles, service requesting orders and service results. Below, a description of the structure elements of the Home Care organization is detailed. Then, in section 4.2, a dynamical usage of the organization is explained, providing different execution scenarios.

4.1 Case Study Organization Structure

This case study is modelled as an organization (HomeCare) within which there are three organizational units (HCServiceUnit, LocationUnit and AlertUnit) each of which represents a group of agents. Each unit is dedicated to home care services, location services or alert services, respectively.

Four kinds of roles can interact in the Home Care example: patient, doctor, family and provider roles. The *Patient role* requests system services. More specifically, it can request home automation services, through the alert service communication with the medical service or the family and more services in their home. The *Doctor role* is specialized in three subroles according to communication with each unit (HCServiceDoctor, LocationDoctor and AlertDoctor). The *Provider role* is in charge of performing services. A provider agent offers home automation, location or alert search services. The provider role is also specialized into HCServiceProvider, LocationProvider and AlertProvider. Finally, the *Family role* provides the advances consultation service. It represents the family in which relatives can check the patient status. As it is a private role, agents are not able to acquire this Family role. Figure 1 shows the Home Care structure, with its organizations/units, roles and relationships with each other.

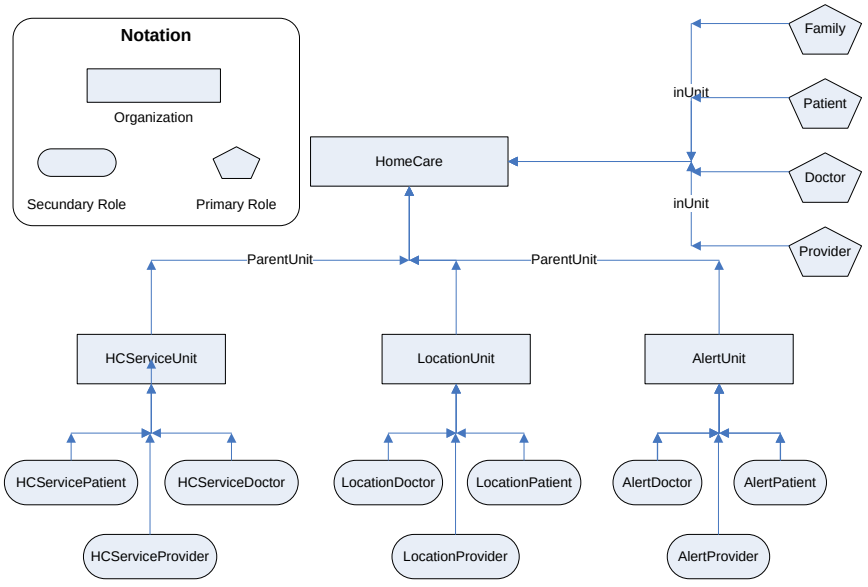


Fig. 1. Home Care structure (units and roles).

The HomeCare organization offers three services: Automation, Location and Alert service. These services are specialized for each unit. A brief description of the profiles of all these services is shown in Table 1.

Table 1. Service Profiles for the HomeCare system.

Profiles of HCServiceUnit		
Service: OnOffLight **UnitID**: HCServiceUnit **Inputs**: idlight: string operation: string	**ProfileID**: OnOffLightPF **ClientRole**: HCServicePatient **Outputs**: [light ok] idlight: string state: string	**Description**: On or off a light. **ProviderRole**: HCServiceProvider **Outputs**: [not ok light] error
Service: LockUnlockAccess **UnitID**: HCServiceUnit **Inputs**: idaccess: string operation: string	**ProfileID**: LockUnlockAccessPF **ClientRole**: HCServicePatient **Outputs**: [access ok] idaccess: string state: string	**Description**: Lock or unlock a access. **ProviderRole**: HCServiceProvider **Outputs**: [not ok acces] error

Profiles of LocationUnit		
Service: SearchPatient **UnitID**: LocationUnit **Inputs**: idhome: string idpatient: string	**ProfileID**: SearchPatientPF **ClientRole**: LocationProvider, LocationDoctor, Family **Outputs**: [patient ok] name: string location: string	**Description**: Search for a patient in their home. **ProviderRole**: LocationProvider **Outputs**: [not in home] error
Service: IdentifyPatient **UnitID**: LocationUnit **Inputs**: idpatient: string	**ProfileID**: SearchPatientPF **ClientRole**: LocationProvider **Outputs**: [patient ok] location: string date: time	**Description**: Identify a patient. **ProviderRole**: LocationProvider **Outputs**: [not ok patient] error
Service: addServPatient **UnitID**: LocationUnit **Inputs**: idpatient: string operation: string	**ProfileID**: AddServPatientPF **ClientRole**: LocationProvider **Outputs**: [patient ok] location: string date: time	**Description**: Add a patient service. **ProviderRole**: LocationProvider **Outputs**: [not ok add patient] error
Profiles of AlertUnit		
Service: SendSms **UnitID**: AlertUnit **Inputs**: sms: string phone: string	**ProfileID**: SendSmsPF **ClientRole**: AlertProvider, AlertDoctor, Family, AlertPatient **Outputs**: [phone ok] idsms: string state: string	**Description**: Send a SMS. **ProviderRole**: AlertProvider **Outputs**: [not ok phone] error
Service: ProcessSms **UnitID**: AlertUnit **Inputs**: sms: string phone: string	**ProfileID**: ProcessSmsPF **ClientRole**: AlertProvider, AlertDoctor, Family, AlertPatient **Outputs**: [sms ok] sms: string phone: string	**Description**: Process a SMS. **ProviderRole**: AlertProvider **Outputs**: [not ok sms] error

All these services have been registered in the SF component of the THOMAS platform. In this example, we have assumed that the Home Care system does not initially have any agent registered as a service provider, nor any agent acting as a patient and nor any agent acting as a doctor. Therefore, this system has initially only been structured as a regulated space in which agents might enter to provide or request

all of those specific services registered in the SF component. Consequently, in the initial state of the system, there is no provider attached to the HomeCare services.

In the following section, different scenarios are considered, in which patient and/or provider and/or doctor agents enter and participate in the system.

4.2 System Dynamics

In this section, the use of THOMAS meta-services in the HomeCare example is detailed. System dynamics are shown through the specification of different scenarios: (i) a Patient is registered; (ii) the patient is registered as a PatientLocation; (iii) new services patients are included; (iv) a doctor is registered; (v) some services are requested; (vi) malicious agents are expulsed; and (vii) a new unit is created.

4.2.1 Patient registering

In this scenario, the process for registering a new Patient is detailed (Fig 2). Once HC1 has been registered as a member of the THOMAS platform, it asks SF which defined services have a profile similar to its own "home care service". This request is carried out using the SF *SearchService* (Fig 2, message 1), in which *HomeCareServiceProfile* corresponds to the profile of the patient search service implemented by HC1.

The SF returns service identifiers that satisfy these search requirements together with a *ranking value* for each service (message 2). *Ranking value* indicates the degree of suitability between a service and a specified service purpose. Then HC1 executes *GetProfile* (message 3) in order to obtain detailed information about the *OpenCloseDoor* service. Service outputs are "service goal" and "profile" (message 4). The *OpenCloseDoor* profile specifies that service providers have to play a *Patient* role within *HCService*. Thus, HC1 requests from the OMS the *AcquireRole* service to acquire this patient role (message 5). *AcquireRole* service is carried out successfully (message 6), because *HCService* is accessible from *Virtual* organization, thus HC1 is registered as a *Patient*.

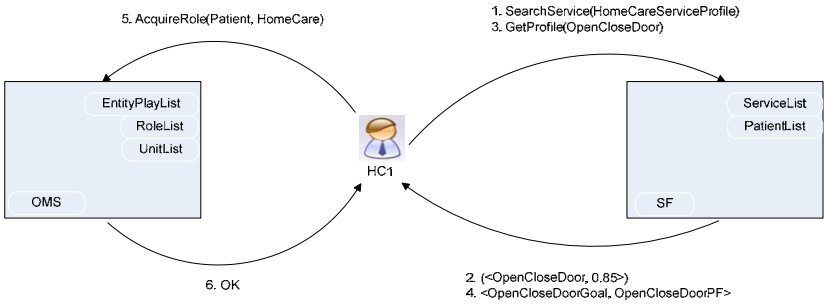


Fig. 2. Example of patient registering.

4.2.2 LocationPatient registering

Once the “patient registering” process has been detailed, the registration of a location patient is illustrated (Fig 3). HC1 is able to provide a search patient in the home care domain. Therefore, it asks SF whether an available service description with a closer profile exists, requesting *SearchService* from SF as before (Fig 3, message 1).

In this case, SF returns both *SearchPatient* and *IdentifyPatient* since these two services are visible within *HomeCare* unit. As indicated in the service result, *IdentifyPatient* service is more appropriate for HC1 functionality. Therefore, HC1 requests information about this service from SF, using *GetProfile* (message 3). The *IdentifyPatient* profile returned (message 4) specifies that service providers must play *LocationPatients* within *LocationUnit*. Then HC1 requests OMS to adopt *LocationPatient* role (message 5). *AcquireRole* service is carried out successfully (message 6), so HC1 agent is registered as a *LocationProvider*.

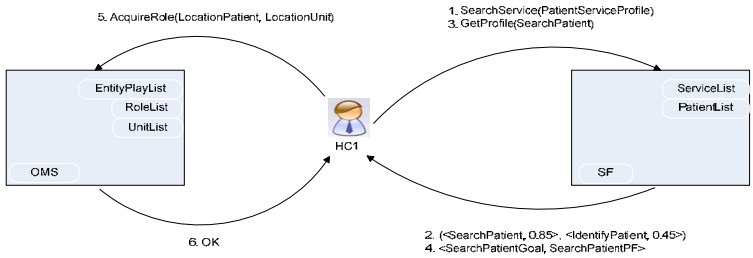


Fig. 3. Example of LocationPatient registering.

4.2.3 Adding new service patient

This section exemplifies how HC2 has already adopted the *LocationPatient* role and HC1 has been registered as a provider of the *SearchPatient* service. HC2 initially asks what the registered implementations of *SearchPatient* service are (Fig 4, message 3). SF provides a list that contains service implementations details (message 4). HC2 decides to employ the same service process as HC1, so it uses *AddServPatient* service in order to request its inclusion as a provider of *SearchPatient* service (Figure 4, message 5).

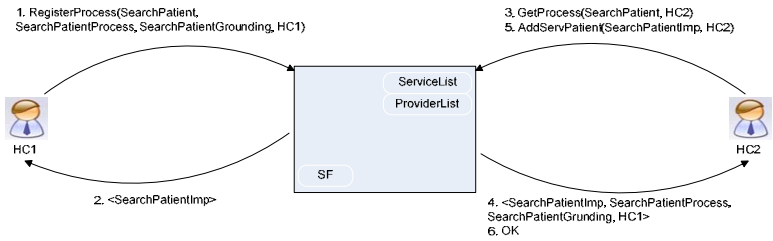


Fig. 4. Example of service implementation and patient registering.

4.2.4 Doctor registering

The following scenario shows the set of service calls for registering new agents as service doctors within the *HomeCare* (Fig 5). A new doctor agent D1, which has already been registered in the THOMAS platform, requests *SearchService* from SF (message 1). As a result, D1 obtains *SearchPatient* service identifier and ranking value (message 2). The *Ranking value* are calculate automatically by the SF. *Ranking value* indicates the degree of suitability between a service and a specified service purpose. Then, D1 employs *GetProfile* (message 3), which specifies that service doctor must play *Doctor* role within *HomeCare* (message 4). Therefore, D1 must acquire *Doctor* role to demand this service (messages 5 and 6). Once D1 plays this doctor role, it employs *GetProcess* service in order to find out who the service providers are and how this service can be requested (message 7). However, there are no providers for the general SearchPatient service (message 8).

Within the *HomeCare* unit, D1 requests *SearchService* again (message 9). In this case, SF returns *IdentifyPatient* services because both services are accessible from *HomeCare* organization. D1 demands the profile of *IdentifyPatient* service (using *GetProfile*, message 11), since this service is more appropriate for its needs. Taking the *IdentifyPatient* profile into account (message 12), D1 requests the adoption of *LocationDoctor* role within *LocationUnit* (message13).

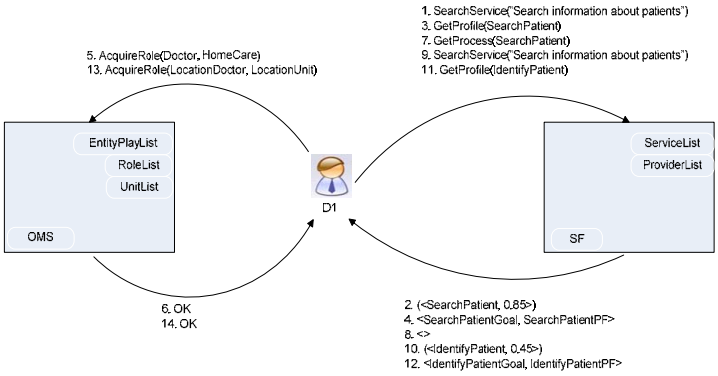


Fig. 5. Example of doctor registering.

4.2.5 Service requesting

This scenario shows how doctor agents make demands for services (Fig 6). Once D1 adopts the doctor role for *SearchPatient* service, it is allowed to demand services from providers. Assuming that D1 wants to make an information search about patients, it should use *GetProcess* service to obtain the implementations of available services and also its provider identifiers (message 1).

An implementation of *SearchPatient* has previously been registered by HC1 and HC2. After comparing providers of *SearchPatient* service returned in message 2, D1 chooses to make a service request from HC1 agent (message 3).

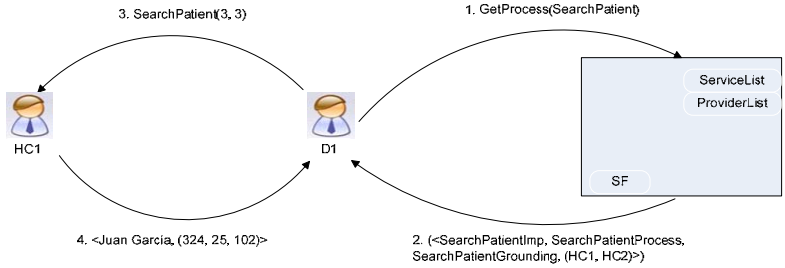


Fig. 6. Example of service requesting.

4.2.6 Agent expulsion

In this scenario, the expulsion of a malicious agent is carried out (Fig7). *Provider* agent detects that different doctor agents (D1 and D2) have registered with the same identifier number. It consults its database and determines that D2 has been employing an identifier number that does not belong to it. D2 is punished for its fraudulent behaviour and is expelled from *HomeCare*. *Provider* requests the expulsion of D2 from OMS employing *Expulse* service (message 1).

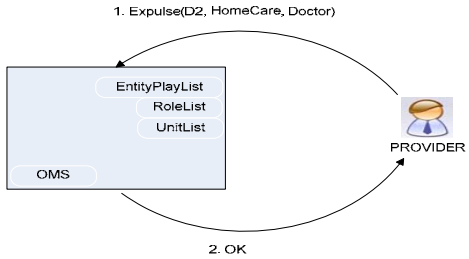


Fig. 7. Example of agent expulsion.

4.2.7 Unit creation

This last scenario illustrates the creation of new units within *HomeCare* (Fig 8). Agent L1 represents a luxury home care company which specializes in luxury services. It is interested in providing information and services very luxurious. This L1 has already adopted the *Provider* role within *HomeCare* unit. However, since the services offered within *LocationUnit* and *AlertUnit* are specialized in location and alert domains, L1 decides to create a new unit (*LuxuryUnit*) within *HomeCare* (Fig 8, message 1). This new unit will be focused on luxury home care. Once the OMS informs L1 about the successful creation of the new unit, L1 defines luxury specific roles and services (messages 3 to 6). Finally, luxury agents would be able to adopt the *LuxuryProvider* role and start offering services to patient agents.

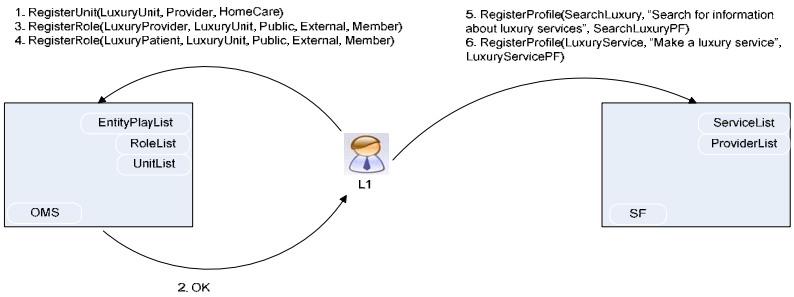


Fig. 8. Example of new unit creation.

After all these scenarios, several agents have joined the THOMAS platform and offer or request services within this system. Table 2 shows the evolution of the *EntityPlayList* content, in which all of the new elements and relationships included due to the execution of these scenarios are emphasized.

Table 2. Final content of OMS internal lists after execution of all scenarios.

EntityPlayList		
Entity	**Unit**	**Role**
Doctor	HomeCare	Doctor
HC1	LocationUnit	LocationPatient
HC2	LocationUnit	LocationPatient
D1	LocationUnit	LocationDoctor
D2	HomeCare	Doctor
L1	LuxuryUnit	LuxuryPatient

5 Conclusions

In the development of real open multi-agent systems, it becomes necessary to have methods, tools and appropriate architectures that can use the concept of agent technology in the development process, and apply decomposition, abstraction and reorganization methods. The THOMAS architecture has allowed us to directly model the organization of a home care environment according to a previous basic analysis, to dynamically and openly define the agent roles, functionalities and restrictions, and to obtain beforehand the service management capabilities (discovery, directory, etc.) within the platform. THOMAS provides us with the level of abstraction necessary for the development of our system, and the set of tools that facilitate its development. Moreover, the proposal aims to instigate the total integration of two promising technologies, that is, multi-agent systems and service-oriented computing. In THOMAS architecture, agents can offer and invoke services in a transparent way from other agents, virtual organizations or entities, plus external entities can interact with agents through the use of the services offered.

A case study example has been applied to home care to illustrate the usage of THOMAS components and services. Also the dynamics applications are developed

with such architecture. In this way, examples of THOMAS service calls have been shown through several scenarios, along with the evolution of different dynamic virtual organizations. THOMAS creates a multi-agent system that facilitates the development of intelligent distributed systems and renders services to dependent person in home care environments by automating certain supervision tasks.

References

1. Anastasopoulos, M., Niebuhr, D., Bartelt, C., Koch, J. & Rausch, A. (2005). Towards a Reference Middleware Architecture for Ambient Intelligence Systems. ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications.
2. Angulo, C., & Tellez, R. (2004). Distributed Intelligence for smart home appliances. Tendencias de la minería de datos en España. Red Española de Minería de Datos. Barcelona, España.
3. Augusto, Juan C., & McCullagh, Paul. Ambient Intelligence: Concepts and Applications. Invited Paper by the International Journal on Computer Science and Information Systems, volume 4, Number 1, pp. 1-28, June 2007.
4. Camarinha-Matos, L. Afsarmanesh, H. TeleCARE: Collaborative Virtual Elderly Care Support Communities. The journal on Information Technology in Healthcare 2004: 2(2): pp. 73-86.
5. Carrascosa, C., Bajo, J., Julian, V., Corchado, J.M. and Botti, V. Hybrid multi-agent architecture as a real-time problem-solving model. Expert Systems With Applications. Volumen 34. Numero 1. pp. 2-17. Elsevier.ISSN:0957-4174 (2008).
6. Carrascosa, C., Giret, A., Julian, V., Rebollo, M., Argente, E., Botti, V.: Service Oriented MAS: An open architecture. In: Actas del AAMAS 2009 (in press, 2009)
7. Corchado, J.M. y Laza, R. (2003). Constructing Deliberative Agents with Case-based Reasoning Technology. International Journal of Intelligent Systems, 18, 1227-1241.
8. Decker, K. and Li, J. (1998). Coordinated hospital patient scheduling. In Proceedings of the 3rd International Conference on Multi-Agent Systems (ICMAS'98) (pp. 104-111). IEEE Computer Society.
9. GTI_IA. A. Giret, V. Julian, M. Rebollo, E. Argente, C. Carrascosa and V. Botti. An Open Architecture for Service-Oriented Virtual Organizations. Seventh international Workshop on Programming Multi-Agent Systems. PROMAS 2009. pp. 23-33. 2009
10. Lanzola, G., Gatti, L., Falasconi, S. and Stefanelli, M. (1999). A Framework for Building Cooperative Software Agents in Medical Applications. Artificial Intelligence in Medicine, 16(3), 223-249.
11. Meunier, J. A. (1999). A Virtual Machine for a Functional Mobile Agent Architecture Supporting Distributed Medical Information. In Proceedings of the 12th IEEE Symposium on Computer-Based Medical Systems (CBMS'99). IEEE Computer Society, Wahington, DC.
12. Miksch, S., Cheng, K. and Hayes-Roth, B. (1997). An intelligent assistant for patient health care. In Proceedings of the 1st international Conference on Autonomous Agents (AGENTS'97) (pp. 458-465). California, USA: ACM, New York.
13. Pecora, F. and Cesta, A. (2007). Dcop for smart homes: A case study. Computational Intelligence, 23 (4), 395-419.
14. Sycara K,Widoffand S, Klusch M, Lu J (1982) Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace. Journal on Autonomous Agents and Multi-Agent Systems.
15. Want, R., Pering, T., Borriello, G., and Farkas, K. Disapearing Hardware. Pervasive Computing, 1(1), (2002).

Auction Robustness through Satisfiability Modulo Theories*

Miquel Bofill[1], Dídac Busquets[2], and Mateu Villaret[1]

[1] Departament d'Informàtica i Matemàtica Aplicada
Universitat de Girona
{miquel.bofill,mateu.villaret}@udg.edu
[2] Institut d'Informàtica i Aplicacions
Universitat de Girona
didac.busquets@udg.edu

Abstract. Solution robustness is a desirable feature when dealing with uncertainty. This issue has rarely been taken into account in the field of auctions, where the goal is to obtain optimal solutions (i.e., maximize the auctioneer's benefit). In this paper we define a notion of robustness for auctions where some resources may become unavailable once the auction has already been cleared. This notion of robustness balances the number of changes needed for repairing the solution and the possible loss of benefit for the auctioneer. In order to obtain robust solutions for auctions, we provide a mechanism based on transporting the concept of supermodel to the setting of weighted Max-SAT. We show that finding a supermodel of a weighted Max-SAT formula amounts to find a model of an SMT (Satisfiability Modulo Theories) formula.

1 Introduction

Auctions are widely used as a resource allocation mechanism, since they enable an efficient distribution of resources amongst the agents requesting them [1]. Most auction mechanisms focus on maximizing the auctioneer's benefit, and assume that once the auction is cleared (i.e., a solution is found), the state of the world cannot change. However, in real applications, this is not the case. In the time between the clearance of the auction and the moment of the allocation of resources taking place, many things can happen. For instance, a machine that has been assigned to a given agent may break down before it starts using it, a winning agent may decide to withdraw its bid because it found a better deal elsewhere, etc. The consequence of such unexpected events is that the solution may then not yield the optimal benefits. Even worse, the solution may not be applicable at all. In such cases, a new solution must be found. To do so, the auction could be repeated, accounting for the new reality. However, the new solution could be completely different from the initial one, meaning that some

* Partially supported by the Spanish Ministry of Science and Innovation through the project SuRoS (ref. TIN2008-04547/TIN)

bids that were winners in the first solution are losers in the new one. Such behavior could be a nuisance for the participating bidders, especially for those that were told their bids were winners but become losers in the repetition of the auction. To avoid such situations, the auctioneer could try to find a solution that is prepared for unexpected events. Ideally, such a solution should be applicable no matter what these events are, although this is not always achievable. Thus, when this happens, the solution should be reparable with the minimum number of changes to the original solution. And, while such a robust solution may be sub-optimal, we are interested in still obtaining a high revenue for the auctioneer.

The issue of bid withdrawal has already been addressed in [2]. But, as far as we know, the problem of resource unavailability has never been taken into account. Thus, in this paper we introduce a new mechanism for finding robust solutions to auctions with a bounded number of allowed breakages in resource availability, a bounded number of repairs in bid assignment, and a minimum guaranteed revenue for the auctioneer:

> If a solution is found, we can guarantee that, for any breakage in resource availability involving at most a resources, the solution can be repaired with at most b changes in bid assignment. In addition, the initial solution and all repaired solutions have a revenue of at least β. We call such a solution an *(a,b,β)-super solution.*

Our starting point is the standard modeling of an auction as a weighted Max-SAT problem [3, 4] with boolean variables to denote whether a bid is winner or loser. In addition, we also use boolean variables representing resource availability. The transformation used to obtain (a, b, β)-super solutions works in the spirit of the one of [5] to obtain (a, b)-supermodels for propositional logic. As in [5], we define a set of breakable variables and a set of repairable variables. Here, the breakage set corresponds to the variables denoting resource availability, and the repair set to those corresponding to bid assignment.

In fact, our mechanism can be applied to any weighted Max-SAT problem, auctions being a particular case. An interesting aspect of our result is that weighted Max-SAT turns out to be not expressive enough to deal with the revenue constraints that we must add in the robust version of the problem. For this reason, we move to the setting of Satisfiability Modulo Theories (SMT) [6] and model the auction as a weighted Max-SMT formula. Hence, for our purposes the transformation of [5] must be adapted to this new setting.

The SMT problem for a theory T is: given a formula F, determine whether there is a model of $T \cup \{F\}$. Hence, an SMT instance is a generalization of a boolean SAT instance in which some propositional variables have been replaced by predicates from the underlying theories. For example, if T denotes the theory of linear (integer or real) arithmetic, then a formula can contain clauses like, e.g., $p \vee q \vee x + 2 \leq y \vee x = y + z$, where p and q are boolean variables and x, y and z integer ones, providing a much richer modeling language than is possible with SAT formulas.

The rest of the paper is organized as follows. In Section 2 we review some related work concerning robustness and auctions. In Section 3 we formally define

the kinds of auctions we consider. In Section 4 we describe an example to show how we look for robust solutions. Section 5 is devoted to the mechanism for finding robust solutions. Finally, in Section 6 we draw some conclusions and devise future work.

2 Related work

Here we briefly review some related work on robust solutions in general, and on robustness and the use of logics in the field of auctions.

2.1 Supermodels

The seminal work on robust solutions for propositional logic formulas is the one of [5], where the notion of *supermodel* is introduced. The complexity for finding such supermodels in several propositional logic fragments has been studied in [7].

Definition 1 (from [5]). *An* (S_1^a, S_2^b)-supermodel *of a boolean formula* F *is a model of* F *such that if we modify the values taken by the variables in a subset of* S_1 *of size at most* $\boldsymbol{a}$ *(breakage), then another model can be obtained by modifying the values of the variables in a disjoint subset of* S_2 *of size at most* $\boldsymbol{b}$ *(repair).*

An (S_1^a, S_2^b)*-supermodel in which the breakage and the repair set are unrestricted is denoted as an* (a, b)*-supermodel.*

The task of finding (a, b)-supermodels is NP-complete. The main idea is to encode the supermodel requirements of a formula F as a new formula F_{SM} whose size is polynomially bounded by the size of F. This new formula F_{SM} has a model if and only if F has an (a, b)-supermodel.

Example 1. The formula $F = p \vee q$ has three models, $\{p, q\}$, $\{\neg p, q\}$ and $\{p, \neg q\}$, which are all $(1, 1)$-supermodels. The encoding F_{SM} for a $(1, 1)$-supermodel of F, according to [5], would be

$$F_{SM} = \overbrace{(p \vee q)}^{\text{original } F} \wedge \overbrace{\Big(\overbrace{(\neg p \vee q)}^{\text{no repair}} \vee \overbrace{(\neg p \vee \neg q)}^{\text{repair } q}\Big)}^{\text{break in } p} \wedge \overbrace{\Big(\overbrace{(p \vee \neg q)}^{\text{no repair}} \vee \overbrace{(\neg p \vee \neg q)}^{\text{repair } p}\Big)}^{\text{break in } q}$$

Note that, for instance, if the satisfying interpretation (model) chosen for F_{SM} is $\{\neg p, q\}$, i.e., $\{p = \mathit{false}, q = \mathit{true}\}$, then $p \vee q$ is satisfied and, moreover, if q switches to *false*, then a new model for $p \vee q$ can be obtained by switching p to *true*. That is, a break in q has a repair on p. The key idea is that the value of the subformula $\neg p \vee \neg q$ under the initial interpretation coincides with the value of $p \vee q$ under the repaired interpretation and, hence, F_{SM} has a model if and only if F has a supermodel. Note also that only the first model $\{p, q\}$ is a $(1, 0)$-supermodel.

The concept of (a, b)-supermodel for propositional logic has been generalized to that of (a, b)-*super solution* in the context of constraint programming in [8].

2.2 Robustness in auctions

There are several works that deal with robustness with respect to potential manipulations of the auction mechanism (such as false-name bids and other types of manipulations). However, this is not the concept of robustness we are interested in. As we have described in Section 1, we focus our research on robustness of the solution to the auction. Some works, such as [9, 10] add the concept of robustness (*fault tolerance*) to *mechanism desing* in order to deal with potential failures in the execution of tasks by the agents, although they use a probabilistic approach and do not consider repairing the solutions. As far as we know, the only work that deals with solution repair in auctions is that of [2]. This work addresses the problem of bid withdrawal (i.e. a bidder that withdraws a winning bid), and, in order to find robust solutions, uses (α, β)-*weighted super solutions* [11], an extension of super solutions [8] that takes into account the breakage probability (α) and the cost of repair (β). Our work is similar, since our approach is also based on supermodels and we look for solutions with a bounded cost. However, we consider the problem of resource unavailability, which is not considered in [2]. Moreover, we are also interested in keeping the number of repairs low, which is only done indirectly (through the cost function) in [2]. In addition, our techniques are completely different because we use the logic framework of weighted Max-SAT and Satisfiability Modulo Theories, while [2] presents an ad-hoc search algorithm to find robust solutions.

Another closely related problem is that of robust knapsack [12] (i.e. a knapsack problem where the weights and/or values of the objects are imprecise). Given that many auction mechanisms can be modeled as a knapsack problem [13], it is reasonable to think that some of the robust approaches to this problem may yield robust solutions to auctions. However, the robustness concept used in the field of knapsack is somehow different to ours, since it does not consider the possibility of repairing a solution. Instead, a robust solution of a knapsack problem with imprecision is such that, on average, performs well regardless of what the actual weights or values of the objects are, in a similar way as the robustness presented in [9, 10].

2.3 Auctions and Logics

Regarding the use of logics in the field of auctions, it has been mainly used to define different bidding languages [14]. There are also some works that use logics as the method for solving the winner determination problem. For instance, Baral et al [15] model the auction using SModels, and use their approach to solve combinatorial auctions and combinatorial exchanges. However, they do not deal with robustness issues. On the other hand, modeling an auction as a weighted Max-SAT formula is a standard problem in Max-SAT benchmarks [4].

3 Auction formalization

There are many types of auctions, depending on several factors, such as the number of items being offered, the number of units for each item, or the way in

which bidders may express their requests, among others [16]. In this work, we restrict our attention to Combinatorial Auctions [17] and, more precisely, to the winner determination problem (clearing algorithm) of the auction. Formally, a combinatorial auction is defined as follows. There is:

- a set of K agents,
- a set of N items or goods, and
- a set of M bids of the form (S_i, p_i), where $S_i \subseteq \{1..N\}$ is the subset of goods (i.e. bundle) requested in bid i, and $p_i \in I\!R^+$ is the value (price) of the bid. We denote by R_j the set of indexes of the bids sent by agent j, $j \in \{1..K\}$.

The goal is to select the winning bids, so that no good is allocated more than once, and the revenue for the auctioneer is maximized:

$$\max \sum_{i=1}^{M} b_i \cdot p_i \quad \text{s.t. } \textstyle\sum_{i|j \in S_i} b_i \leq 1, \forall j \in \{1..N\}$$

where b_i is a boolean variable indicating whether bid i is winner or loser.

Additionally, one may introduce other constraints, such as forcing all items to be allocated (which would imply replacing the $\leq$ of the previous equation by an equality), guaranteeing that all agents win at least (or at most) a given number of bids ($\sum_{i \in R_j} b_i \geq Q,\ \forall j \in \{1..K\}$, where Q is the minimum number of winning bids per agent – or maximum if using $\leq Q$), etc.

Regarding the bidding language, in the rest of the paper we assume that bidders use an OR-language [14], that is, each bidder submits a list of bids (pairs of bundle and price), and it is interested in winning any number of the bids sent to the auctioneer. However, the approach we present may work with any other bidding language (XOR, OR-of-XOR, ...), as long as the needed restrictions (as mentioned in the previous paragraph) are added.

4 Running example

In order to illustrate our approach, we present an example that we will use in the forthcoming sections to explain each of the steps needed to find robust solutions to auctions. For the sake of simplicity, we use single-item bundles, that is, each bid requests only one item. Moreover, we impose the constraint that each agent must win at least one of the bids it sent.

Example 2. Assume we have 3 agents and 4 goods (or resources). In the following table we indicate the price of each single-item bid of each agent:

	1	2	3	4
1	10	15	-	-
2	-	5	10	20
3	15	-	-	10

Thus we have the following list of bids:

$$[\underbrace{(1,10),(2,15)}_{\text{first agent's bids}},\underbrace{(2,5),(3,10),(4,20)}_{\text{second agent's bids}},\underbrace{(1,15),(4,10)}_{\text{third agent's bids}}]$$

We define the boolean variables g_1, g_2, g_3 and g_4 to represent the availability of the corresponding goods, and the boolean variables $b_i, i \in \{1..7\}$ to indicate whether bid i is winner or loser. Then, assuming that the only possible source of breakages is resource availability, we define the breakage set as being $S_1 = \{g_1, g_2, g_3, g_4\}$. Then, the repair set is $S_2 = \{b_1, b_2, b_3, b_4, b_5, b_6, b_7\}$, since a break in a resource may imply that a winning bid becomes loser and, eventually, other assignments can be reconsidered in order to improve the auctioneer's revenue under the new circumstances.

Assume that we look for *robust solutions* where one break may occur and each possible break must be repairable with at most four changes. Assume, moreover, that we want that whatever the break is, the revenue of the initial solution and of the repaired solution is at least 30. This will correspond to a $(1, 4, 30)$-super solution (see Definition 2 of Subsection 5.2).

The optimal solution to this auction without considering robustness would be to set as winning bids the second, the fourth, the fifth and the sixth bids, i.e.,

$$b_1 = 0, b_2 = 1, b_3 = 0, b_4 = 1, b_5 = 1, b_6 = 1, b_7 = 0,$$

which means assigning good 2 to the first agent, goods 3 and 4 to the second agent and good 1 to the third agent. For the sake of readability we use the notation such as 2456 to indicate which are the winning bids of a solution. With this solution, the auctioneer would have a revenue of 60.

However, this optimal solution is not a $(1, 4, 30)$-super solution: if good 2 became unavailable (break), the only alternative for the first agent would be good 1, but this is already allocated to the third agent; this would imply finding also an alternative for the third agent, which would be good 4, but this good is allocated to the second agent. Thus, repairing the breakage of good 2 would imply modifying two winning bids (b_2 to b_1 and b_6 to b_7) and unassigning one winning bid (b_5), meaning five repairs (as shown in boldface):

$$\mathbf{b_1 = 1}, \mathbf{b_2 = 0}, b_3 = 0, b_4 = 1, \mathbf{b_5 = 0}, \mathbf{b_6 = 0}, \mathbf{b_7 = 1},$$

which is more than the four allowed repairs. Note that for each bidder, choosing a new winning bid may imply two repairs (one to set the initially winning bid to 0, and in case he had no other winning bids, another one to set one of its losing bids to 1).

This auction has 9 feasible solutions (i.e., solutions satisfying the constraints on bid incompatibility and minimum number of winning bids per agent), which are the following: 1247, 137, 1347, 147, 246, 2456, 247, 2467 and 256. Within these solutions, only three of them are $(1, 4, 30)$-super solutions: 246, 2467 and 247. Next we go through the details of solution 2467, i.e.,

$$b_1 = 0, b_2 = 1, b_3 = 0, b_4 = 1, b_5 = 0, b_6 = 1, b_7 = 1$$

which has a revenue of 50 units for the auctioneer. For this solution, the four possible breakages can be repaired as follows:

1. $g_1 = 0$. The repair is $b_6 = 0$, being the new solution 247, and the revenue 35.
2. $g_2 = 0$. The repair is $b_1 = 1, b_2 = 0, b_6 = 0$, being the new solution 147 and the revenue 30.
3. $g_3 = 0$. The repair is $b_4 = 0, b_5 = 1, b_7 = 0$, being the new solution 256 and the revenue 50.
4. $g_4 = 0$. The repair is $b_7 = 0$, being the new solution 246 and the revenue 40.

It can be seen that all repairs have a revenue of at least 30 and the number of repairs is not greater than 4. Moreover, in the third case there is not even loss in the revenue. We let the reader check that solutions 246 and 247 are also $(1, 4, 30)$-super solutions, with a revenue of 40 and 35, respectively. However, since the revenue of solution 2467 is higher, this would be the optimal $(1, 4, 30)$-super solution to this auction.

As for the rest of feasible solutions, some of them (147 and 1247) are $(1, 4, \_)$-super solutions, meaning that they can be repaired with at most 4 changes, but not $(1, 4, 30)$-super solutions, since they do not satisfy that the solution and its repairs have a revenue of at least 30. In particular, solution 147 has a revenue of 30, but one of its repairs has a revenue of only 25 (when good 3 becomes unavailable, the second agent must be assigned good 2, which corresponds to a low value bid). Similarly, solution 1247 has a revenue of 45, but it also fails in the revenue of repairs, since one of them has again a revenue of 25.

Finally, some solutions (137, 1347, 2456 and 256) are not even $(1, 4, \_)$-super solutions, since they do need more than 4 changes in order to repair some of the breakages. This is the case of the optimal solution without robustness (2456), as we have seen a few paragraphs above.

5 Mapping auctions to supermodels

5.1 Auctions as weighted Max-SAT problems

An auction A as defined in Section 3 can be modeled as a weighted Max-SAT formula F^A as follows. Let $g_1, \ldots, g_N$ be a set of boolean variables representing the availability of each of the N goods, and let $b_1, \ldots, b_M$ be a set of boolean variables representing whether each of the M bids is winner or loser.

1. *Resource availability.* For each bid j of the form (S_j, p_j), where $S_j = \{i_1, \ldots, i_{n_j}\} \subseteq \{1..N\}$, we state

$$b_j \rightarrow g_{i_1} \wedge \cdots \wedge g_{i_{n_j}}$$

to indicate that, whenever bid j is accepted, then all goods it requests must be available[3].

[3] These constraints are only needed to deal with resource unavailability, i.e., they do not appear in standard (non-robust) auctions.

2. *Bid incompatibility.* For each pair of bids i and j (with $i \neq j$) such that $S_i \cap S_j \neq \emptyset$, we state
$$\neg b_i \vee \neg b_j$$
to indicate incompatibility between bids requesting the same good.
3. *Minimum winning bids.* For each set $R_k = \{i_1, \ldots, i_{n_k}\}$ corresponding to the bids of agent k, we state
$$b_{i_1} \vee \cdots \vee b_{i_{n_k}}$$
to indicate that at least one bid of each agent must be accepted. Note that this restriction can change, or even disappear, depending on the type of bidding language used in the auction (OR, XOR, OR-of-XOR, ...) and also on the constraints about the number of winning bids per agent.
4. *Bid's value.* For each bid i, we add a unit clause
$$(b_i, p_i)$$
indicating that if bid i is not accepted, then there is a loss of revenue of p_i.

The conjunction of the previous constraints defines a weighted Max-SAT problem for the auction.

5.2 Robust auctions as weighted Max-SMT problems

Here we show how a weighted Max-SAT formula F^A defining an auction A can be transformed into a weighted Max-SMT formula defining a robust version of the former. In particular, we describe how to obtain a Max-SMT formula F^A_{SM} such that F^A_{SM} has a model if and only if A has an (a, b, β)-super solution. Some of the proofs are omitted due to lack of space.

Definition 2. *An* (a, b, β)-super solution *of an auction is a (maximal revenue) solution for the auction such that, if* $\boldsymbol{a}$ *goods become unavailable (breakage), then another solution can be obtained by changing at most* $\boldsymbol{b}$ *bids from winner to loser or vice-versa (repair) and, moreover, the solution and all possible repaired solutions have a revenue of at least* β*.*

The following definition generalizes the one of [5] to weighted Max-SAT.

Definition 3. *An* (S^a_1, S^b_2, β)*-supermodel of a weighted Max-SAT formula* F *is a model of* F *such that if we modify the values taken by the variables in a subset of* S_1 *of size at most* $\boldsymbol{a}$ *(breakage), another model can be obtained by modifying the values of the variables in a disjoint subset of* S_2 *of size at most* $\boldsymbol{b}$ *(repair) and, moreover, the solution and all possible repaired solutions have a cost of at most* β*.*

Lemma 1. *An auction* A *with* N *goods and* M *bids has an* (a, b, β)*-super solution if and only if the weighted Max-SAT formula* F^A *has an* (S^a_1, S^b_2, β')*-supermodel where* $S_1 = \{g_1, \ldots, g_N\}$*,* $S_2 = \{b_1, \ldots, b_M\}$ *and cost (i.e. loss of revenue)* $\beta' = \left(\sum_{i=1}^{M} p_i\right) - \beta$*.*

Now we show how to construct a weighted Max-SMT formula F_{SM} from a weighted Max-SAT formula F, such that F has an (S_1^a, S_2^b, β)-supermodel if and only if F_{SM} has a model.

The construction of F_{SM}. Let

$$F = C \wedge W$$

be a weighted Max-SAT formula, where C denotes the set of mandatory constraints and W denotes the set of weighted, non-mandatory constraints[4]. For the sake of simplicity, we will assume that W consists only of unary clauses of the form (b, w), where b is a boolean variable and w is a weight. Note that any Max-SAT formula can be transformed into an equisatisfiable one fulfilling this requirement by reification, i.e., by replacing any weighted constraint (G, w) such that G is not unary by $(G \leftrightarrow b) \wedge (b, w)$.

Now, assuming that

$$W = (b_1, w_1) \wedge \cdots \wedge (b_k, w_k)$$

we introduce a set of integer variables $i_1, \ldots, i_k$ and define

$$L = \bigwedge_{j \in 1..k} (b_j \rightarrow i_j = 1) \wedge (\neg b_j \rightarrow i_j = 0)$$

Let S^n denote the set of all (possibly empty) subsets of a set S whose size is at most n, and let S^{n+} denote the set of all non-empty subsets of a set S whose size is at most n. Moreover, let $F_{\overline{S}}$ denote a boolean formula F where all occurrences of variables in the set S have been flipped (i.e., negated).

We define

$$B_{\overline{S}} = \sum_{j \in 1..k} \begin{cases} i_j \cdot w_j & \text{if } b_j \in S \\ (1 - i_j) \cdot w_j & \text{if } b_j \notin S \end{cases}$$

where S is a set of boolean variables. We denote by B the particular case $B_{\overline{\emptyset}} = \sum_{j \in 1..k} (1 - i_j) \cdot w_j$, corresponding to the cost of the unsatisfied clauses in W.

Finally, we define

$$F_{SM} = C \wedge W \wedge L \wedge (B \leq \beta) \wedge \bigwedge_{S \in S_1^{a+}} \left(\bigvee_{T \in (S_2 \setminus S)^b} (C_{\overline{S \cup T}} \wedge (B_{\overline{S \cup T}} \leq \beta)) \right)$$

Note that, due to L and the constraints of the form $B \leq \beta$, this formula is not plain SAT: it falls into SAT modulo the quantifier-free fragment of the (first-order) linear arithmetic theory. The main lemma and theorem follow here.

Lemma 2. *A weighted Max-SAT formula F has an (S_1^a, S_2^b, β)-supermodel if and only if the weighted Max-SMT formula F_{SM} has a model.*

[4] We talk about constraints instead of clauses since our transformation does not require the formula to be in CNF format.

This lemma follows the spirit of the result of [5], but adding costs. Here, the key idea is that $B_{\overline{S \cup T}}$, gives us the cost of the unsatisfied clauses in W if the variables in $S \cup T$ were to change their value with respect to the initial solution. Note that the variables in S represent the breakage variables and the ones in T represent the repair variables and, hence, $S \cup T$ denotes the set of variables that are going to change their value. Notice also that the sets S and T are disjoint, since it makes no sense to repair a broken variable.

Theorem 1. *An auction A has an (a, b, β)-super solution if and only if the weighted Max-SMT formula F^A_{SM} has a model.*

Proof. Let $F^A = C \wedge W$ denote the weighted Max-SAT formula obtained from A as explained in Subsection 5.1, where C denotes the set of mandatory constraints introduced in points 1, 2, and 3, and W denotes the set of non-mandatory unit clauses of point 4. Let $S_1 = \{g_1, \ldots, g_N\}$, $S_2 = \{b_1, \ldots, b_M\}$ and $\beta' = \left(\sum_{i=1}^{M} p_i\right) - \beta$. By Lemma 2, F^A has an (S^a_1, S^b_2, β')-supermodel if and only if F^A_{SM} has a model. Therefore, by Lemma 1, A has an (a, b, β)-super solution if and only if F^A_{SM} has a model. □

Observe that, since a and b are constants, the increase of size of F^A_{SM} w.r.t. the size of F^A is polynomially bounded, namely, it is $\mathcal{O}(n^{a+b})$ larger than F^A, where n is the number of variables of F^A.

5.3 The whole story

Here we take the auction A of Example 2 and briefly describe how to model it as a robust auction. We begin by modeling the auction as a Max-SAT problem as explained in Subsection 5.1, which gives us $F^A = C \wedge W$ with

$$\begin{aligned} C = \; & (b_1 \rightarrow g_1) \wedge (b_2 \rightarrow g_2) \; \wedge (b_3 \rightarrow g_2) \wedge (b_4 \rightarrow g_3) \wedge (b_5 \rightarrow g_4) \; \wedge \\ & (b_6 \rightarrow g_1) \wedge (b_7 \rightarrow g_4) \; \wedge (\neg b_1 \vee \neg b_6) \wedge (\neg b_2 \vee \neg b_3) \wedge (\neg b_5 \vee \neg b_7) \wedge \\ & (b_1 \vee b_2) \wedge (b_3 \vee b_4 \vee b_5) \wedge (b_6 \vee b_7) \\ W = \; & (b_1, 10) \wedge (b_2, 15) \wedge (b_3, 5) \wedge (b_4, 10) \wedge (b_5, 20) \wedge (b_6, 15) \wedge (b_7, 10) \end{aligned}$$

Now observe that the sum of the costs of the non-mandatory weighted clauses is $10+15+5+10+20+15+10 = 85$. Then, as stated by Lemma 1, in order to A have a $(1, 4, 30)$-super solution, we must look for a $(S^1_1, S^4_2, 85-30)$-supermodel of F^A, i.e., a $(S^1_1, S^4_2, 55)$-supermodel of F^A, where $S_1 = \{g_1, g_2, g_3, g_4\}$ and $S_2 = \{b_1, b_2, b_3, b_4, b_5, b_6, b_7\}$. Finally, according to Lemma 2, this amounts to find a model of the weighted Max-SMT formula

$$F^A_{SM} = C \wedge W \wedge L \wedge (B \leq 55) \wedge \bigwedge_{S \in S^{1+}_1} \left(\bigvee_{T \in (S_2 \setminus S)^4} (C_{\overline{S \cup T}} \wedge (B_{\overline{S \cup T}} \leq 55)) \right)$$

as described in Subsection 5.2, where

$$L = \bigwedge_{j \in 1..7} (b_j \rightarrow i_j = 1) \wedge (\neg b_j \rightarrow i_j = 0)$$

$$B = (1 - i_1) \cdot 10 + (1 - i_2) \cdot 15 + (1 - i_3) \cdot 5 + (1 - i_4) \cdot 10 + \\ (1 - i_5) \cdot 20 + (1 - i_6) \cdot 15 + (1 - i_7) \cdot 10$$

Note that S_1^{1+} denotes the non-empty subsets of S_1 with at most one element, i.e., the singletons $\{g_1\}, \{g_2\}, \{g_3\}$ and $\{g_4\}$. And, since S_1 and S_2 are disjoint, we have that $(S_2 \setminus S)^4 = S_2^4$, i.e., the (possibly empty) subsets of S_2 of size at most 4. Due to lack of space we do not develop $C_{\overline{S \cup T}}$ and $B_{\overline{S \cup T}}$.

6 Conclusions

In this paper we have presented a mechanism for obtaining robust solutions to auctions. This issue has rarely been considered in the field of auctions, with only a few exceptions [2, 9, 10]. However, we think that robustness is a key issue when dealing with real world applications, where uncertainty is almost always present. In particular, we have focused on the possibility of some of the resources becoming unavailable once the auction has already been cleared. Thus, we provide a mechanism to proactively look for solutions that can be easily repaired when such unexpected events happen.

We have presented a notion of robustness that balances the number of allowed repairs when a break occurs and the loss of revenue for the auctioneer, by defining what we call (a, b, β)-super solutions. This allows the auctioneer to choose the more convenient values of each parameter, depending on how conservative or risk seeking its strategy is.

Our mechanism is based on the modeling of an auction as a weighted Max-SAT formula. However, since SAT does not allow to easily encode formulas with arithmetic operations, needed to achieve robustness, we have moved the problem to the richer logical framework of Satisfiability Modulo Theories.

Let us mention that state-of-the art SMT solvers have a rich input language, and it is not necessary (neither convenient) to translate any formula to CNF in order they can read it. In fact, we can feed an SMT solver directly with a formula such as F_{SM} (as described in Subsection 5.2) contrarily to what is done in [5], where all formulas are translated to CNF. It is worth noting that we have not considered the option of translating our transformed formula into a linear program since, on the one hand, SMT solvers dealing with the theory of linear arithmetic already apply a (modified) simplex algorithm and, on the other hand, such a translation would imply a flattening of the structure of the problem (of which the SMT solvers can eventually take revenue) and the addition of many new variables. Moreover, the richness of the SMT language allows us to easily add new constraints when needed. Nevertheless, we have left as future work a peformance comparison with Operational Research tools.

Some experiments have been carried out with Yices [18], a weighted Max-SMT solver. At present only toy examples have been checked. In the future we plan to test our transformation with realistic benchmarks from the Combinatorial Auctions Test Suite (CATS) [19].

Finally, we want to acknowledge the fruitful discussions and comments we've had with the rest of the SuRoS project team.

References

1. McMillan, J.: Selling spectrum rights. Journal of Economic Perspectives **8**(3) (1994) 145–162
2. Holland, A., O'Sullivan, B.: Robust solutions for combinatorial auctions. In: ACM Conf. on Electronic Commerce. (2005)
3. Larrosa, J., Heras, F., de Givry, S.: A logical approach to efficient max-sat solving. Artif. Intell. **172**(2-3) (2008) 204–233
4. Argelich, J., Li, C.M., Manyà, F., Planes, J.: The first and second Max-SAT evaluations. J. on Satisfiability, Boolean Modeling and Computation **4** (2008) 251–278
5. Ginsberg, M.L., Parkes, A.J., Roy, A.: Supermodels and robustness. In: Proc. of AAAI'98. (1998) 334–339
6. Sebastiani, R.: Lazy satisability modulo theories. J. on Satisfiability, Boolean Modeling and Computation **3**(3-4) (2007) 141–224
7. Roy, A.: Fault tolerant boolean satisfiability. J. Artificial Intelligence Research **25** (2006) 503–527
8. Hebrard, E., Hnich, B., Walsh, T.: Super solutions in constraint programming. In: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Lecture Notes in Computer Science. Springer (2004) 157–172
9. Ramchurn, S.D., Mezzetti, C., Giovannucci, A., Rodriguez, J.A., Dash, R.K., Jennings, N.R.: Trust-based mechanisms for robust and efficient task allocation in the presence of execution uncertainty. J. of Artificial Intelligence Research **35** (2009) 119–159
10. Porter, R., Ronen, A., Shoham, Y., Tennenholtz, M.: Fault tolerant mechanism design. Artificial Intelligence **172**(15) (2008) 1783 – 1799
11. Holland, A., O'Sullivan, B.: Weighted super solutions for constraint programs. In: Proc. of AAAI'05. (2005) 378–383
12. Yu, G.: On the max-min 0-1 knapsack problem with robust optimization applications. Operations Research **44** (1996) 407–415
13. Kelly, T.: Generalized Knapsack Solvers for Multi-unit Combinatorial Auctions: Analysis and Application to Computational Resource Allocation. LNAI **3435** (2005) 73–86
14. Nisan, N.: Bidding and allocation in combinatorial auctions. In: Proc. of ACM Conference on Electronic Commerce. (2000) 1–12
15. Baral, C., Uyan, C.: Declarative specification and solution of combinatorial auctions using logic programming. In: Proc. of LPNMR'01, LNCS. Volume 2173. (2001) 186–199
16. Wurman, P.R., Wellman, M.P., Walsh, W.E.: A parametrization of the auction design space. Games and Economic Behavior **35**(1-2) (2001) 304 – 338
17. Cramton, P., Shoham, Y., Steinberg, R., eds.: Combinatorial Auctions. MIT Press (2006)
18. Dutertre, B., de Moura, L.: The Yices SMT solver. Tool paper available at http://yices.csl.sri.com/tool-paper.pdf (August 2006)
19. Leyton-Brown, K., Pearson, M., Shoham, Y.: Towards a universal test suite for combinatorial auction algorithms. In: Proc. of ACM Conference on Electronic Commerce. (2000) 66–76

Self-Adaptive MAS for Biomedical Environments

Juan F. De Paz, Sara Rodríguez, Javier Bajo and Juan M. Corchado

Departamento de Informática y Automática, Universidad de Salamanca
Plaza de la Merced s/n, 37008, Salamanca, España
{fcofds, srg, jbajope, corchado}@usal.es
Department of Computer Science and Automation, University of Salamanca Plaza de la Merced s/n, 37008, Salamanca, Spain

Abstract. The application of information technology in the field of biomedicine has become increasingly important over the last several years. This paper presents an intelligent dynamic architecture for knowledge data discovery in biomedical databases. The core of the system is a type of agent that integrates a novel strategy based on a case-based planning mechanism for automatic reorganization. This agent proposes a new reasoning agent model, where the complex processes are modeled as external services. The agents act as coordinators of Web services that implement the four stages of the case-based planning cycle. The multi-agent system has been implemented in a real scenario to classify leukemia patients. The classification strategy includes services to analyze patient's data, and the results obtained are presented within this paper.

Keywords: Multiagent Systems, Case-Based Reasoning, microarray, Case-based planning

1. Introduction

The continuous growth of techniques for obtaining cancerous samples, specifically those using microarray technologies, provides a great amount of data. Microarray has become an essential tool in genomic research, making it possible to investigate global genes in all aspects of human disease [8]. Expression arrays [9] contain information about certain genes in a patient's samples. These data have a high dimensionality and require new powerful tools. Usually, existing systems are focused on working with very concrete problems or diseases, with low dimensionality for the data, and it is very difficult to adapt them to new contexts for diagnosing different diseases. This research presents an entirely new perspective that focuses on the concept of Intelligent Organizations, proposing an architecture capable of modeling biomedical organizations through multi-agent systems to analyze biomedical data.

This paper presents an innovative solution to model decision support systems in biomedical environments, based on a multi-agent architecture which allows integration with Web services and incorporates a novel planning mechanism that makes it possible to determine workflows based on exising plans and previous results. The Multiagent System centers on obtaining a self-adaptive biomedical organizational model, making it possible to represent laboratory workers within a virtual

environment and the interactions that take place, in order to carry out daily classification tasks. The core of system is a CBP-BDI (Case-based planning) (*Belief* Desire Intention) agent [3] specifically designed to act Web services coordinator, making it possible to reduce the computational load for the agents in the organization and expedite the classification process. CBP-BDI agents [3] make it possible to formalize systems by using a new planning mechanism that incorporates graph theory and Bayesian networks as a reasoning engine to generate plans. The system was applied to case studies, consisting of the classification of leukemia patients and brain tumors from microarrays, and the multiagent system developed incorporates novel strategies for data analysis and microarray data classification. Microarray has become an essential tool in genomic research, making it possible to investigate global gene expression in all aspects of human disease [8].

The next section describes the main characteristics of the proposed multiagent system and briefly explains its components. Section 3 presents a case study consisting of a distributed multi-agent system for cancer detection scenarios. Finally section 4 presents the results and conclusions obtained.

2. Multiagent System for Expresion Analysis

Nowadays, having software solutions at one's disposal that enforce autonomy, robustness, flexibility and adaptability of the system to develop is completely necessary. The dynamic agents organizations that auto-adjust themselves to obtain advantages from their environment seems a more than suitable technology to cope with the development of this type of systems. The integration of multi-agent systems with SOA (*Service Oriented* Architecture) and Web Services approaches has been recently explored [14]. Some developments are centred on communication between these models, while others are centred on the integration of distributed services, especially Web Services, into the structure of the agents. Ricci et al. [15] have developed a java-based framework to create SOA and Web Services compliant applications, which are modelled as agents. Communication between agents and services is performed by using what they call “artifacts” and WSDL (Web Service Definition Language). We have used the FUSION@ architecture [12] as a reference, which not only provides communication and integration between distributed agents, services and applications.

The approach presented in this paper is an organizational model for biomedical environments based on a multi-agent dynamic architecture that incorporates agents with skills to generate plans for analysis of large amounts of data. The core of the system is a novel mechanism for the implementation of the stages of CBP-BDI mechanisms through Web services that provides a dynamic self-adaptive behaviour to reorganize the environment. Moreover, the system provides communication mechanisms that facilitate integration with SOA architectures. The multiagent system was initially designed to model the laboratory environments oriented to the processing of data from expression arrays. To do this, the system defined specific agent types and services. The agents act as coordinators and managers of services, while the services are responsible for carrying out the processing of information by providing replication

features and modularity. Agents are available to run on different types of devices, so different versions were created to suit each one. The types of agents are distributed in layers within the system according to their functionalities, thus providing an organizational structure that includes an analysis of the information and management of the organization, and making it possible to easily add and eliminate agents from the system. The agent layers constitute the core and define a virtual organization for massive data analysis, as can be seen in Figure 1. Figure 1 shows four types of agent layers:

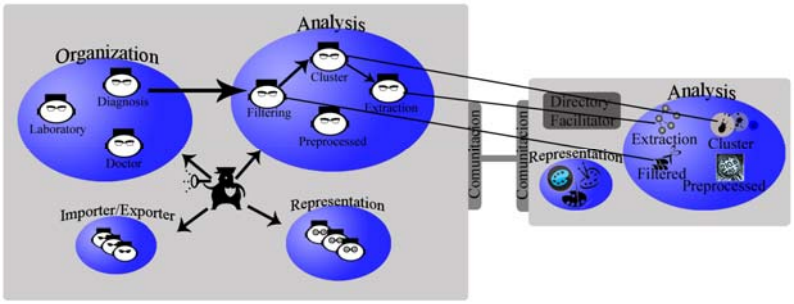


Fig. 1. Multiagent System Architecture

- Organization: The agents will be responsible for conducting the analysis of information following the CBP-BDI [3] reasoning model. The agents from the organizational layer should be initially configured for the different types of analysis that will be performed. Because these analyses vary according to the available information and the search results.
- Analysis: The agents in the analysis layer are responsible for selecting the configuration and the flow of services that best suit the problem to solve. They communicate with Web services to generate results. The agents of this layer follow the CBP-BDI [3] reasoning model. The workflow and configuration of the services to be used is selected with a Bayesian network and graphs, using information that corresponds to the previously executed plans. The agents at this layer are highly adaptable to the case study to which is applied. Specifically, the microarray case study includes those agents that are required to carry out the expression analysis, as shown in figure 1.
- Representation: These agents are in charge of generating the tables with the classification data and the graphics for the results.
- Import/Export: These agents are in charge of formatting the data in order to adjust them to the needs of agents and services.
- The Controller agent manages the agents available in the different layers of the multiagent system. It allows the registration of agents in the layers, as well as their use in the organization.

On the other hand, the services layer is divided into two groups:

- Analysis Services: The analysis services are services used by analysis agents for carrying out different tasks. The analysis services include services for pre-processing, filtering, clustering and extraction of knowledge. Figure 1 illustrates

how these services are invoked by the analysis layer agents in order to carry out the different tasks corresponding to microarray analysis.

- Representation Services: They generate graphics and result tables.

Within the services layer, there is a service called Facilitator Directory that provides information on the various services available and manages the XML file for the UDDI (Universal Description Discovery and Integration). To facilitate communication between agents and services the architecture integrates a communication layer that provides support for the FIPA-ACL and SOAP protocols.

Figure 1 shows the connections between the diagnosis agent (in the organization layer) with the agents in the analysis layer and the services. The connections represent a plan. A diagnosis incorporates a filtering process, carried out by an analysis agent that selects the sequence of services for the plan. Then, a clustering agent selects the optimum service. Finally, the knowledge extraction obtains the relevant probes.

Nowadays, there exist different possibilities to services planning and composition. One of the most important is services composition using HTN (Hierarchical Task Network) and HTN planners as SHOP2 [20]. These systems don't provide a planning mechanism that make use of past experiences, so they have a lack of adaptation an learning abilities. Another techniques are based on Quality of Service [21] that make use of heuristics to obtain an optimum composition. However, the quality of each of the services is not independent of the others.

2.1. Coordinator CBP-BDI Agent

The coordinator agent is the core of the system, since provides the ability for self-organization. The agents in the organization layer have the capacity to learn from the analysis carried out in previous procedures. They adopt the model of reasoning CBP, a specialization of case-based reasoning (CBR) [2]. CBP is the idea of planning as remembering [3]. In CBP, the solution proposed to solve a given problem is a plan, so this solution is generated taking into account the plans applied to solve similar problems in the past [13]. The problems and their corresponding plans are stored in a plans memory. A plan P is a tuple $<S,B,O,L>$, S is the set of plan actions, O is an ordering relation on S allowing to establish an order between the plan actions, B is a set that allows describing the bindings and forbidden bindings on the variables appearing in P, L is a set of casual links.

The CBP-BDI agents stem from the BDI model [16] and establish a correspondence between the elements from the BDI model and the CBP systems. The BDI model adjusts to the system requirements since it is able to define a series of goals to achieve based on the information that has been registered with regards to the world. Fusing the CBP agents together with the BDI model and generating CBP-BDI agents makes it possible to formalize the available information, the definition of the goals and actions that are available for resolving the problem, and the procedure for resolving new problems by adopting the CBP reasoning cycle.

The CBP-BDI agent type presented in this paper acts as coordinator of services. The terminology used is the following: The environment M and the changes that are

produced within it, are represented from the point of view of the agent. Therefore, the world can be defined as a set of variables that influence a problem faced by the agent

$$M = \{\tau_1, \tau_2, \cdots, \tau_s\} \text{ with } s < \infty \tag{1}$$

The beliefs are vectors of some (or all) of the attributes of the world taking a set of concrete values

$$B = \{b_i / b_i = \{\tau_1^i, \tau_2^i, \cdots, \tau_n^i\}, n \leq s \quad \forall i \in N\}_{i \in N} \subseteq M \tag{2}$$

A state of the world $e_j \epsilon E$ is represented for the agent by a set of beliefs that are true at a specific moment in time t. i represents a belief of the N.

Let $E=\{e_j\}_{j \epsilon N}$ set of status of the World if we fix the value of t then

$$e_j^t = \{b_1^{jt}, b_2^{jt}, \cdots b_r^{jt}\}_{r \in N} \subseteq B \quad \forall j, t \tag{3}$$

The desires are imposed at the beginning and are applications between a state of the current world and another that it is trying to reach

$$d : \underset{e_0}{E} \underset{\rightarrow}{\rightarrow} \underset{e^*}{E} \tag{4}$$

Intentions are the way that the agent's knowledge is used in order to reach its objectives. A desire is attainable if the application *i*, defined through n beliefs exists:

$$i : \underset{(b_1, b_2, \cdots, b_n, e_0)}{BxBx^{n)}\cdots xBxE} \underset{\rightarrow}{\rightarrow} \underset{e^*}{E} \tag{5}$$

In our model, intentions guarantee that there is enough knowledge in the beliefs base for a desire to be reached via a plan of action. We define an agent action as the mechanism that provokes changes in the world making it change the state,

$$a_j : \underset{e_i}{E} \underset{\rightarrow}{\rightarrow} \underset{a_j(e_i)=e_j}{E} \tag{6}$$

Agent plan is the name we give to a sequence of actions that, from a current state e_0, defines the path of states through which the agent passes in order to reach the other world state.

$$p_n : \underset{e_0}{E} \underset{\rightarrow}{\rightarrow} \underset{p_n(e_0)=e_n}{E} \tag{7}$$

$$p_n(e_0) = e_n = a_n(e_{n-1}) = \cdots = (a_n \circ \cdots \circ a_1)(e_0) \quad p_n \equiv a_n \circ \cdots \circ a_1$$

Based on this representation, the CBP-BDI coordinator agents combine the initial state of a case, the final state of a case with the goals of the agent, and the intentions with the actions that can be carried out in order to create plans that make it possible to reach the final state. The actions that need to be carried out are services, making a plan an ordered sequence of services. It is necessary to facilitate the inclusion of new services and the discovery of new plans based on existing plans. Services correspond to the actions that can be carried out and that determine the changes in the initial problem data. Each of the services is represented as a node in a graph. The presence of an arch that connects to a specific node implies the execution of a service associated with the end node. Figure 2 provides a graphical representation of a service plans. As shown, the first graph has only one path and contains nodes that are not

connected. The path defines the sequence of services from the start node until the end node. The plan described by the graph is defined by the sequence $(S_7 \, o \, S_5 \, o \, S_3 \, o \, S_1)(e_0)$. e_0 represents the original state that corresponds to Init, which represents the initial problem description e_0. Final represents the final state of the problem e^*.

CBP-BDI agents use the information contained in the cases in order to perform different types of analyses. As previously explained, an analysis assumes the construction of the graph that will determine the sequence of services to be performed. The construction process for the graph can be broken down into a series of steps that are explained in detail in the following sub-sections:

1. Generate the directed graph with the information from the different plans.
2. Generate a TAN (Tree Augmented Naive Bayes) classifier for the cases with the best and worst output respectively, using the Friedman-Godsmidtz [17] algorithm.
3. Calculate the execution probabilities for each service with respect to the classifier generated in the previous step.
4. Adjust the connections from the original graph according to a metric.
5. Construct the graph

2.1.1. Constructing a directed graph

The different plans are represented in the graphs. The plans represented in graphical form are joined to generate one directed graph that defines the new plans based on the minimization of a specific metric. For example, given the graphs shown in figure 2, a new graph is generated that joins the information corresponding to both graphs.

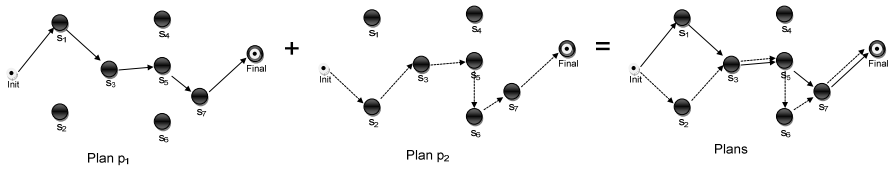


Fig. 2. Composition of the graphs

The dual connection of the nodes is indicated only to represent the existence of a connection between the two graphs, although it is not actually necessary to represent more than one connection per arc. Each of the arcs in the graph for the plans has a corresponding weight according to which it is possible to calculate the new route to be executed. This value is estimated based on the efficiency of the plans recovered as indicated in section 2.1.4. When constructing the graph of plans, the weights are estimated based on the existing plans by applying a bayesian network. The entry data to the bayesian network is broken down into the following elements: Plans with a high efficiency are assigned to class 1 and plans with a low efficiency are assigned to class 0. The Bayesian network is calculated for each of the classes according to the recovered plans, following the Friedman-Goldsmidtz [17] algorithm.

2.1.2. TAN classifier

The TAN classifier is constructed based on the plans recovered that are most similar to the current plan, distinguishing between efficient and inefficient plans to generate

the model. Thus, by applying the Friedman-Goldsmidtz [17] algorithm, the two classes that are considered are efficient and inefficient. The Friedman-Goldsmidtz [17] algorithm makes it possible to calculate a Bayesian network based on the dependent relationships established through a metric. The metric proposed by Friedman is defined as follows:

$$I(X;Y \mid Z) = \sum_{x \in X} \sum_{y \in Y} \sum_{z \in Z} P(x,y,z) \cdot \log\left[\frac{P(x,y \mid z)}{P(x \mid z) \cdot P(y \mid z)}\right] \quad (8)$$

Based on the previous metric, the maximal tree is constructed.

2.1.3. Services Probabilities

Once the TAN model has been calculated for each of the classes, we proceed to calculate the probability of execution for each of the services. These probabilities influence the final value of the weights assigned to the arcs in the graph. The probabilities are calculated according to the TAN model. Assuming that the set of random variables can be defined as $U = \{X_1, X_2, \ldots, X_n\}$, we can assume that the variables are independent. The probabilities are represented by $P(x_i \mid \pi_{x_i})$ where x_i is a value of the variables X_i and $\pi_{x_i} \in \prod_{X_i}$ where π_{x_i} represents one of the parents for the node X_i. Thus, a Bayesian network B, defines a single set probability distribution over U given for

$$P(X_1, X_2, \ldots, X_n) = P(X_n \mid X_{n-1}, \ldots, X_1) \cdot P(X_{n-1}, \ldots, X_1) =$$

$$\prod_{i=1}^{n} P(X_n \mid X_{n-1}, \ldots, X_1) = \prod_{i=1}^{n} P(X_i \mid \Pi_{X_i})$$

2.1.4. Considering the connections

Using the TAN model, we can define the probability that a particular number of services may have been executed for classes 1 and 0. This probability is used to determine the final value for the weight with regards to the quality of the plans recovered. Assuming that the probability of having executed service i for class c is defined as follows $P(i,c)$ the weight of the arcs is defined according to the following formula. The function has been defined in such a way that the plans of high quality are those with values closest to zero.

$$c_{ij} = P(j,1) \cdot I(i,j,1) \cdot t_{ij}^{1} - P(j,0) \cdot I(i,j,0) \cdot t_{ij}^{0} \quad (9)$$

$$t_{ij}^{1} = \frac{\sum_{p \in G_{ij}^{1}, s \in G^{1}} (1 - (q(p) - \min(q(s)))) + 0.1}{\#G_{ij}^{1}} \quad (10)$$

$$t_{ij}^{0} = \frac{\sum_{p \in G_{ij}^{0}, s \in G^{0}} q(p) - \min(q(s)) + 0.1}{\#G_{ij}^{0}} \quad (11)$$

where:

- $I(i,j,c)$ is the probability that service i for class c is executed before of service j
- $P(j,c)$ is the probability that service j for class c is executed. The value is obtained based on the Bayesian network defined in the previous step.
- G^{s}_{ij} is the set of plans that contain an arc originating in j and ending in i for class s.

- G^s is the set of plans for class *s*.
- *q(p)* is the quality of plan p that also defined the execution time for the plan. The significance depends on the measure of optimization in the initial plan.
- $\#G^s_{ij}$ the number of elements in the set.
- c_{ij} is the weight for the connection between the start node *j* and the end node *i*.

2.1.5. Graph construction

Once the graph for the plans has been constructed, the minimal route that goes from the start node to the end node is calculated. In order to calculate the shortest/longest route, the Dijkstra algorithm is applied since there are implementations for the order *n*log n*. To apply this algorithm, it is necessary to add to each of the edges the absolute value of the edge with a higher negative absolute value, in order to remove from the graph those edges with negative values. The route defines the new plan to execute and depends on the measure to maximize or minimize.

3. Case Study: A Decision Support System for Patients Diagnosis

The multiagent architecture presented in this paper has been used to develop a decision support system for the classification of leukemia and brain tumors patients, and three case studies were established. The first case study uses data from patients suffering from leukemia and focuses on the classification of the type of leukemia. The second case study also analyzes the data from leukemia patients, but in this case focuses on the type of CLL leukemia and attempts to classify the patients in the three existing subtypes. Finally, the goal of the third case study is to classify patients based on the type of brain tumor. The data for leukemia patients was obtained with a HG U133 plus 2.0 chip and corresponded to 212 patients affected by 5 different types of leukemia (ALL, AML, CLL, CML, MDS) [19]. The second case study also used the HG U133 plus 2.0 chip. Finally, third case study [18] used data from the Affymetrix U95Av2 GeneChips including 4 different types of brain tumors [18].

3.1. Services Layer

The services implement the algorithms that allows the analysis expression of the microarrays [1] [19]. There are four types of services:

Preprocessing Service: This service implements the RMA (Robust Multi-array Average) [5] algorithm and a novel control and errors technique. During the Control and Errors phase, all probes used for testing hybridization are eliminated.

Filtering Services: Eliminate the variables that do not allow classification of patients by reducing the dimensionality of the data. Three services are used for filtering: (i) Variables with low variability have similar values for each of the individuals, so they are not significant for the classification process. (ii) All remaining variables that follow a uniform distribution are eliminated. The contrast of assumptions followed uses the Kolmogorov-Smirnov [6] test. (iii) The linear

correlation index of Pearson is calculated and correlated variables are removed. (iv) Delete the probes which don't have significative changes in the density of individuals.

Clustering Service: It addresses both the clustering and the association of a new individual to the group more appropriate. The service used is the ESOINN (Enhanced self-organizing incremental neuronal network) [4]. Additional services in this layer are the Partition around medoids (PAM) [10] and dendrograms [11]. Classification is carried out bearing in mind the similarity of the new case using the naive bayes.

Knowledge Extraction Service: The extraction of knowledge technique applied has been CART (Classification and Regression Tree) [7] algorithm.

3.2. Agents Layer

The agents in the analysis layer implement the CBP reasoning model and, for this, select the flow for services delivery and decide the value of different parameters based on previous plans made. A measure of efficiency is defined for each of the agents to determine the best course for each phase of the analysis process. In the analysis layer, at the stage Preprocessed only a service is available. The efficiency is calculated by the deviation in the microarray. At the stage of filtering, the efficiency of the plan p is calculated by the relationship between the proportion of probes and the resulting proportion of individuals falling ill.

$$e(p) = \frac{s}{N} + \frac{i'}{I} \tag{12}$$

Where s is the final number of variables, N is the initial number of probes, i' the number of misclassified individuals and I the total number of individuals. In the phase of clustering and classification the efficiency is determined by the number of misclassified individuals. Finally, in the process of extracting knowledge at the moment, efficiency is determined by the number of misclassified individuals.

In the organization layer, the diagnosis agent chooses the agents for the expression analysis [1]. The diagnosis agent establishes the number of plans to recover from the plans memory for each of the agents and the agents to select from the analysis layer.

4. Results and Conclusions

This paper has presented the a self-adaptive multiagent architecture and its application to three real problems. The characteristics of this novel architecture facilitate a organizational-oriented approach where the dynamics of a real scenario can be captured and modelled into CBP-BDI agents. The tests were oriented both to evaluate the efficiency and the adaptability of the approach. The first experiment consisted of evaluating the services distribution system in the filtering agent for the case study that classified patients affected by different types of leukemia. According to the identification of the problem described in table 1, the filtering agent selected the plans with the greatest efficiency, considering the different execution workflows for the services that are in the plans. Table 1 shows the efficiency obtained for the service workflows that provided the best results in previous experiences. The values in the

table indicates the application sequence for the services within the plan, a blank cell indicates that a service is not invoked for that specific plan.

Based on the plans shown in table 1, a new plan is generated following the procedures indicated in section 2.1. The filtering agent in the analysis layer selects the configuration parameters between a specific set of pre-determined values, when it has been told to explore the parameters. Otherwise, for a specific plan, it selects the values that have provided better results based on the measure of the previously established efficiency (12). If there is no plan with all the services that are going to be used, it selects the plan with the greatest efficiency that contains the greatest number of services equal to the current plan for selecting the different parameters.

Table 1. Efficiency of the plans

Variability (z)	Uniform (α)	Correlation (α)	Cutoff	Efficiency	Class
1	2	3	4	0.1401	1
1	2		3	0.1482	1
1				0.1972	0
	1		2	0.1462	1
		1		0.2036	0
	1			0,1862	0
			1	0,1932	0
		1	2	0,186	0

Table 2. Efficiency of the plans

Case study	Variability (z)	Uniform (α)	Correlation (α)	**Cutoff**	Efficiency
Leukemia	**1**	**2**	**3**		**0.1277**
CLL Leukemia	1	2	3	**4**	0.1701
Brain	1		2		0,1532

Once the service distribution process and the selection of parameters for a specific case study have been evaluated, it would appear convenient to evaluate the adaption of this mechanism to case studies of a different nature. To do so, we once again recover the plans with the greatest efficiency for the different workflows and case studies, and proceed to calculate the Bayesian network and the set of probabilities associated with the execution of services as mentioned in sections 2.1.2 and 2.1.3. Once the graph plans have been generated, a more efficient plan is generated according to the procedures indicated in section 2.1.5, with which we can obtain the plan that best adjusts to the data analysis. Table 2 shows the plans generated by the filtering process that best adjusts to the different case studies.

In Figure 3 it is possible to observe the performance of the agents at the organization and analysis layers. 11 plans were conducted based on manual planning and the results were compared with the automatic analysis provided by the multiagent system. In the manual planning a human expert configures the service's parameters, such as if the RMA will use interquantile normalization, or the sequence to execute the services. Each of the agents of the organization layer selects the agents from the analysis layer and, each of these agents in turn selects the services and configuration parameters. The different kinds of agent from the analysis layer can be seen at the bottom of Figure 3 (the name of the agents from the organization layer is indicated in

the right of the graphics). In each chart the efficiency measure used is shown. The surface for the CBP-BDI agent is the highest efficiency according to the definitions.

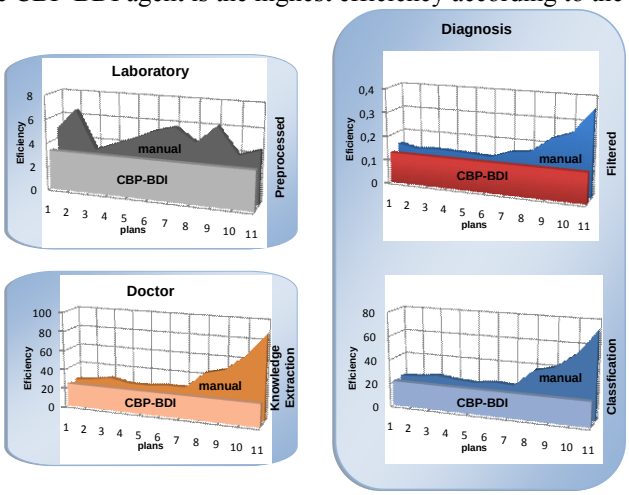


Fig. 3. Performance Comparison between the manual and the automatic planning

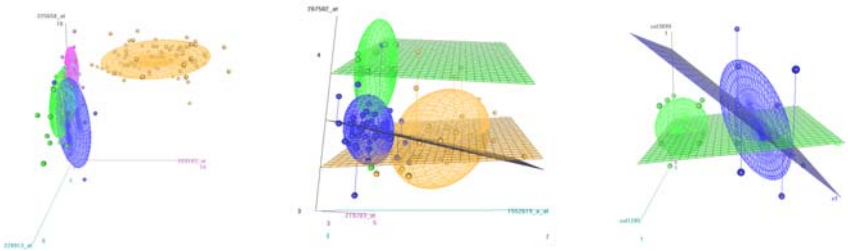


Fig. 4. Patients with CLL leukemia, CLL leukemia subtypes and patients with anaplastic and oligodendrogliomas tumors.

With regards to the classification process, we were able to obtain promising results for each of the case studies. As shown in figures 4a 4b 4c, the probes recovered by the knowledge extraction agent are those that provide the relevant information that makes it possible to classify new individuals. In the first image, we can see the 3 probes that best characterize the patients with CLL leukemia. In the second image, we can see the 3 subtypes of leukemia. Finally, the last image represents the patients with anaplastic and oligodendroglioma tumors. The multi agent system simulates the behavior of experts working in a laboratory, making it possible to carry out a data analysis in a distributed manner, as normally done by experts. The system distributes the functionality among Web services, automatically calculates the expression analysis and allows the classification of patients from the microarray data. Our approach improves the performance provided by the manual procedure for selecting workflow analyses.

References

[1] Lander,E. et al: Initial sequencing and analysis of the human genome. Nature. 409 860-921 (2001)

[2] Kolodner, J.: Case-Based Reasoning. Morgan Kaufmann. (1993)

[3] Glez-Bedia, M. and Corchado, J.: A planning strategy based on variational calculus for deliberative agents. Computing and Information Systems Journal. 10 (1) 2-14 (2002)

[4] Furao, S., Ogura, T. and Hasegawa, O.: An enhanced self-organizing incremental neural network for online unsupervised learning. Neural Networks. 20 893-903 (2007)

[5] Irizarry, R., Hobbs, B., Collin, F., Beazer-Barclay, Y., Antonellis, K., Scherf, U. and Speed, T.: Exploration, Normalization, and Summaries of High density Oligonucleotide Array Probe Level Data. Biostatistics. 4 249-264 (2003)

[6] Brunelli, R.: Histogram Analysis for Image Retrieval. Pattern Recognition. 34 1625-1637 (2001)

[7] Breiman, L., Friedman, J., Olshen, A. and Stone, C.: Classification and regression trees. Wadsworth International Group. Belmont, California. (1984)

[8] Quackenbush, J.: Computational analysis of microarray data Nature Review Genetics. 2 (6) 418-427 (2001)

[9] Affymetrix http://www.affymetrix.com/support/technical/datasheets/hgu133arrays_datasheet.pdf

[10] Saitou, N. and Nie, M.: The neighbor-joining method: A new method for reconstructing phylogenetic trees. Molecular Biology and Evolution. 4 406-425 (1987)

[11] Kaufman, L. and Rousseeuw, P.: Finding Groups in Data: An Introduction to Cluster Analysis. Wiley Series in Probability and Statistics. (1990)

[12] Tapia, D.I., Rodriguez, S., Bajo, J. and Corchado, J.M.: FUSION@, A SOA-Based Multi-agent Architecture. International Symposium on Distributed Computing and Artificial Intelligence, Advances in Soft Computing. 50 99-107 (2008)

[13] Corchado, J.M., Bajo, J., De Paz, Y. and Tapia, D.I.: Intelligent Environment for Monitoring Alzheimer Patients, Agent Technology for Health Care. Decision Support Systems. 44 (2) 382-396 (2008)

[14] Ardissono, L., Petrone, G. and Segnan, M.: A conversational approach to the interaction with Web Services. Computational Intelligence, Blackwell Publishing. 20 693-709 (2004)

[15] Oliva, E., Natali, A., Ricci, A., and Viroli, M.: An Adaptation Logic Framework for {J}ava-based Component Systems. Journal of Universal Computer Science. 14 (13) 2158-2181 (2008)

[16] Bratman, M.: Intention, Plans and Practical Reason. Harvard U.P., Cambridge. (1987)

[17] Friedman, N., Geiger, D. and Goldszmidt, M.: Bayesian Network Classifiers. Machine Learning. 29 131-163 (1997)

[18] Nutt, C.L., Mani, D.R., Betensky, R.A., Tamayo, P., Cairncross, J.G., Ladd, C., Pohl, U., Hartmann, C., McLaughlin, M.E., Batchelor, T.T., Black, P.M., von Deimling, A., Pomeroy, S.L., Golub, T.R., Louis, D.N.: Gene expression-based classification of malignant gliomas correlates better with survival than histological classification. Molecular Biology and Genetics. 63 (7) 1602-1607 (2003)

[19] Corchado, J.M., De Paz, J.F., Rogríguez, S. and Bajo, J.: Model of experts for decision support in the diagnosis of leukemia patients. Artificial Intelligence in Medicine. 46 (3) 179-200 (2009)

[20] Sirin E., Parsia B., Wu, D., Hendler J., Nau D.: HTN planning for Web Service composition using SHOP2. Web Semantics: Science, Services and Agents on the World Wide Web, 1 (4) 377-396 (2004)

[21] Ko, J.M., Kim, C.O., Kwon, I.H.: Quality-of-service oriented web service composition algorithm and planning architecture. The Journal of Systems and Software.81 2079–2090 (2008)

Agreement Patterns

Carlos A. Iglesias[1]**, Mercedes Garijo[1],
José I. Fernández-Villamor[1], and José J. Durán[2]

[1] Depto. Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid
[2] Centro para las Tecnologías Inteligentes de la Información y sus Aplicaciones (CETINIA), Universidad Rey Juan Carlos

Abstract. Agreement Patterns have been defined for improving communication in software and services development, as well as for providing a practitioner oriented approach for reusing existing agreement technologies. This article presents the notion of agreement patterns, their structure and some of the first identified examples.

1 Introduction

One of the envisioned aspects of Future Internet is the Internet of Services, which is conceived as [1] an open service delivery platform, which goes beyond the client-server model of service delivery.

In an Internet of Services with a vast and changing landscape of service providers and consumers, the fulfillment of agreements for services is a key issue, which involves notions such as trust, coordination or organization. These notions are the basis for the understanding and implementation of artificial social systems [2], and have largely addressed in fields such as multi-agent systems [3,4,5], p2p computing [6], service computing [7,8,9], autonomic computing [10], social psychology [11], sociobiology or social neuroscience.

Agreement technologies [12] (AT) is a recent discipline which collects this multidisciplinary research and can be defined as *the technologies for the practical application of knowledge to the automated fulfillment of agreements.* Agreement technologies do not dictate the underlying technologies (objects, components, agents, services, ...), but are focused on the formalization of knowledge structures, protocols, algorithms and expertise that contribute to the establishing of agreements in an open dynamic environment.

Nevertheless, there has not been defined yet a common vocabulary or patterns for sharing and reusing previous experiences in the application of agreements to social-inspired software integration. This article pretends to bridge the gap through the use of software patterns for agreement analysys and reuse, which are called *agreement patterns.*

The rest of the article is organized as follows. Section 2 reviews some of the related work in patterns in the SOA and multi-agent community. Then, section 3

** The first author has been partially supported by Germinus XXI (Grupo Gesfor) through the project RESULTA.

presents a classification scheme for agreement patterns. Section 4 introduces some examples of identified agreement patterns. Finally, section 5 draws out the conclusions of this research work and the future work.

2 Background and Related Work

Software design patterns [13] can be defined as "*a technique for achieving widespread reuse of software architecture. They capture the static and dynamic structures and collaborations of components in successful solutions to problems that arise when building software in domains like business data processing, telecommunications, graphical user interfaces, databases and distributed communication software. Patterns aid the development of reusable components and frameworks by expressing the structure and collaboration of participants in a software architecture at a level higher than source code or object-oriented design models.*".

Patterns can be represented in an informal way using natural language, UML, pattern languages or ontologies [14]. The informal representation of software patterns usually follows the canonical (or so-called Alexandrian) form [15], which identified the following elements:

- **Name:** a meaningful name that provides a vocabulary for discussing.
- **Alias:** an alternative name to the pattern.
- **Problem:** a statement of the problem and the goals it wants to reach.
- **Context:** the preconditions under which the problem and its solution seem to recur.
- **Forces:** a description of the relevant forces and constraints and how they interact with one another and with the goals. Considerations to be taken into account to select a solution for a problem.
- **Solution:** static relationships and dynamic rules describing how to realized the desired outcome.
- **Examples:** one or more sample applications of the pattern which illustrate its application. Known occurrences of the pattern which help in verifying that the pattern is a proven solution to a recurring problem.
- **Resulting context:** the state or configuration of the system after the pattern has been applied.
- **Rationale:** a justification of the pattern, explaining how and why it works, and why it is "good".
- **Related patterns:** compatible patterns which can be combined with the described pattern.

Based on this previous research, **Agreement patterns** are defined as *software patterns that facilite software components coordination through the fulfillment of agreements*. Agreements patterns include all kind of agreements, both explicit ones (e.g. negotiation) and tacit ones (e.g. organization).

There are related works for defining design patterns in the areas of multi-agent systems and Service Oriented Computing (SOC).

Agent-Oriented Patterns have been defined for sharing multi-agent system development experiences. Oluyomi [16,17] presents an agent pattern classification scheme based on two dimensions: stages of the agent-oriented software development and tasks in each stage of development. At each stage or level of development (analysis, multi-agent architecture, agent architecture, multi-agent implementation), the framework identifies the attributes of that level of abstraction, in order to classify these patterns. In addition, Oluyomi proposes to refine the canonical pattern form for defining an Agent-Oriented Pattern Template Structure, which adds more granularity depending on the pattern type (agent internal architecture structural, interactional or strategic patterns, etc.). Some of the patterns identified by Oluyomi, whose classification scheme includes other approaches, can be considered agreement patterns. The main differences between her classification and the one proposed in this article is that Oluyomi's classification is agent oriented (agent oriented development phase, agent architecture, etc.), while the one proposed here is independent of the technology to be used, although implementation examples can be presented with different technologies. In addition, agreements are not a key concept in Oluyomi's classification scheme as in our proposal. Future work will provide a mapping of the agreement related patterns classified by Oluyomi onto our classification scheme.

In the area of Service Oriented Architecture (SOA), *SOA patterns* have been defined [18,19,20]. For example, Erl [18] classifies patterns for architecturing services, service compositions, service inventories and service oriented enterprise. Rotem-Gal-Oz [19] describes patterns for Message Exchange, Service Interaction, Service Composition, Structural, Security and Management. SOA patterns [20] provide high level architectural patterns, which do not detail yet agreement issues.

Inside the SOC community, the GRAAP Working Group (Grid Resource Allocation and Agreement Protocol WG) has defined the specification Web Services Agreement [8], which is particularly interesting for this research. The purpose of the specification is the definition of a Web Services protocol for establishing agreements defined in XML. The specification covers the specification of agreement schemas, agreement template schemas and a set of port types and operations for managing the agreement life cycle. This specification defines an agreement as *an agreement between a service consumer and a service provider specifies one or more service level objectives both as expressions of requirements of the service consumer and assurances by the service provider on the availability of resources and/or service qualities. An agreement defines a dynamically-established and dynamically-managed relationship between parties. The object of this relationship is the delivery of a service by one of the parties within the context of the agreement. The management of this delivery is achieved by agreeing on the respective roles, rights and obligations of the parties.* An agreement is characterized by its name, context and terms.

The OASIS Reference Architecture for SOA [9] is *an abstract realization of SOA, focusing on the elements and their relationships needed to enable SOA-based systems to be used, realized and owned.* The reference architecture defines

three primary viewpoints: *business via services* that captures what SOA means for people using it to conduct business, *realizing service oriented architectures* deals with the requirements for constructing a SOA; and *owning service oriented architectures* addresses issues involved in owning and managing a SOA. The notion of agreement is included in several ways in the architecture, as an organizational concept (constitution) or as a formalization of a relationship (business agreement and contract).

These two initiatives,OASIS RA and WS-Agreement are compatible and complementary of our proposal, since they provide a modeling reference architecture as well as a language for describing the identified patterns boiling down to the implementation level. This integration will be included in future publications.

3 Classification Scheme for Agreement Patterns

The purpose of this section is an early identification of a multi-dimensional classification scheme for agreement patterns (figure 1) which makes easier for software developers to select the right pattern to apply, as well as the basis for its cataloging.

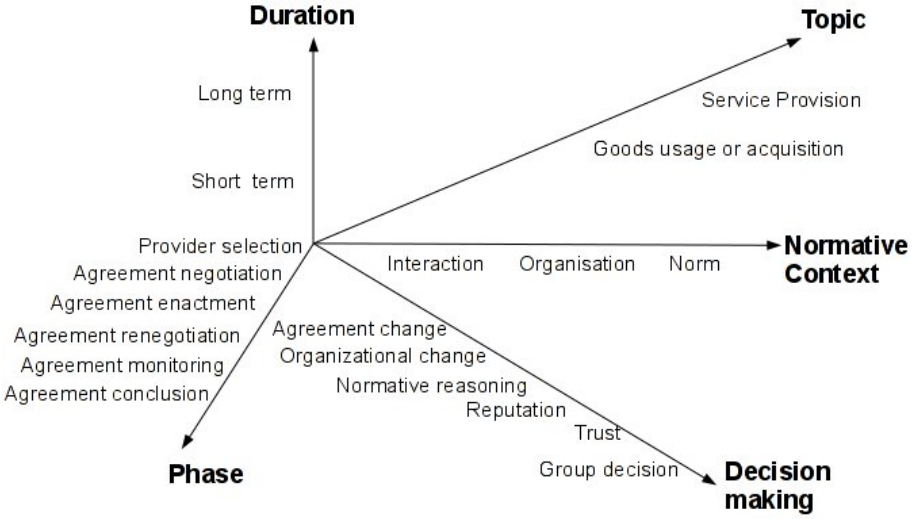


Fig. 1. Classification scheme for Agreement Patterns

The identified dimensions characterize an agreement from several properties of the agreement itself, such as topic (purpose of the agreement), duration, and phase (also so-called state in WS-Agreement); the organizational environment where the agreement takes place, and the decision making cognitive task of the agreement stakeholders.

One pattern can have more than one value for each one of the dimensions, and can be classified only to the relevant dimensions.

The dimensions of the classification scheme are:

- **Duration:** if the agreement is reached for a short or long term. The duration of the agreement has organizational implications, as well as its properties. Pěchouček [21] distinguishes between *alliances* for long-term collaboration agreements and *coalitions* for short-term collaboration agreements. In a similar way, other authors such as Camarinha [22] identifies *long term strategic alliances* and *goal-oriented networks*, based on this dimension.
- **Topic:** the goal of the agreement can be [5], among others, the obtention of goods, the usage of goods or services, the provision of a service or the definition of social rules.
- **Phase:** this dimension defines the agreement phase where the pattern can be applied. The phases considered are provider selection, agreement fulfillment, agreement renegotiation, agreement monitoring and agreement conclusion. *Provider selection* consists of the selection of the best available provider before establishing an agreement, which can be based on trust and reputation techniques and can involve interactions such as bidding or contract net protocol (CNP) [23]. *Agreement negotiation* is the process of defining establishing the conditions of the agreement, respecting the normative context and using negotiation techniques such as argumentation. *Agreement enactment* is the commitments of the agreement are effective. For example, in the call-by-agreement interaction method [24], once the agreement for action is establish, then the actual enactment of the action is requested. In the CNP, the successful bidders are informed that they are now contractors for a task together with this task specification. *Agreement renegotiation* is the phase of changing the conditions of a previously negotiated agreement, because of reasons such as environmental changes, expiration of the agreement or the coming up of a renewal option. *Agreement monitoring* is the process of reviewing the established agreement conditions while the agreement is being enacted. In the area of service computing, it is usually referred as Service Level Agreement monitoring. Finally, *Agreement conclusion* is the process of concluding an agreement by any party, which can be because the agreement is not being enacted according to the negotiated terms, or because one or more parties desires to conclude it.
- **Normative context:** the normative context [24] determines the rules of the game, such as interaction patterns, norms and organization structures.
- **Decision making:** this dimension collects cognitive patterns followed by the parties of the agreement. Some of the main subcategories are *trust and reputation* [25], *group decision* [26], *organizational change* [24], reasoning about the impact of the organization evolution in the current negotiated agreements and in the agreement procedures; *normative reasoning* [24] for interpreting the normative context for a specific case; *agreement change*, reasoning about the improvement of the negotiated agreements as well as its interpretation in specific situations.

The dimensions are interrelated, and one value in one dimension can constraint the values of other dimensions. For example, the topic (goods acquisition) can restrict the valid values of normative context, being interaction (bidding, contract-net protocol, ...) or organizational structure (electronic institution, ...).

4 Examples

This section provides examples of some of the first identified agreement patterns, whose pattern structure follows the canonical form extended with the classification scheme defined in section 3. The decision making process is shown in a reasoning diagram which follows CommonKads notation [27] as illustrated in 2.

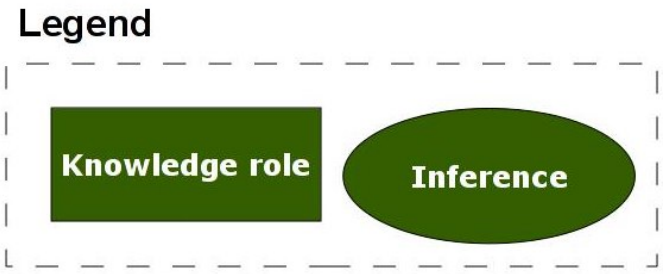


Fig. 2. Inference diagram legend

4.1 ProviderRating Pattern

Selection of the most satisfactory service providers for a specific service demand is known as the *Service Selection* problem [28,29].

This pattern collects the pattern of rating available service providers in order to select the one with highest rating. Rating-based mechanisms are simple and effective [30], if consumers have similar preferences.

Name ProviderRating
Duration Short term (specific project)
Phase Provider Selection
Decision making Reputation, Trust
Problem Choosing a service provider, having access to other users' rating.
Context Public rating of services.
Forces A comparison of the experience-based service-selection techniques can be found in [31].
Solution Service provider selection is based on the rating given to service providers using reputation [29] or trust [32] techniques.

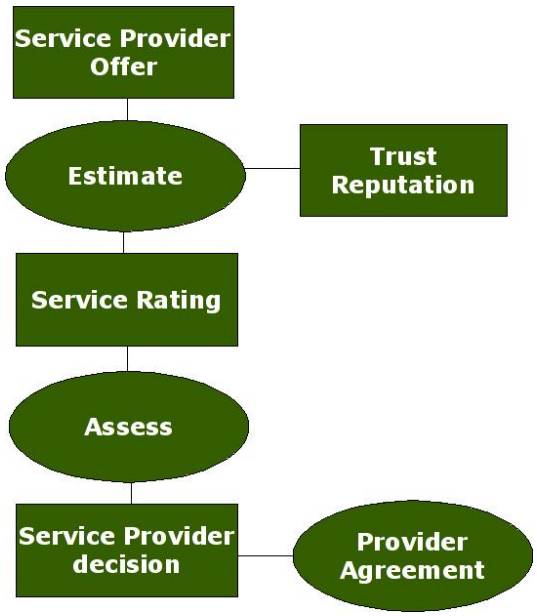


Fig. 3. Provider Rating Reasoning Diagram

Examples E-Commerce [33]
Resulting context An agreement is fulfilled.
Related Patterns Portability

4.2 Portability Pattern

This pattern collect the problematic of changing of provider without service interruption, which involves monitoring the current QoS and the estimation of other Service Provider offer. When the decision of changing is taking, this pattern involves the negotiation of breaking the existing contract and agree a new contract with the new provider.

Usually, service providers try to prevent this portability, defining some *customer retention* policy to improve *customer loyalty*.

Name *Portability*
Alias *Service switching [34]*
Problem A potentially better service offer is available and the customer wants to break the service provision agreement with the current service provider and to establish a new agreement with a new provider, without disruption of the service.
Context There are several available service providers that offer a better service conditions than the current provider.

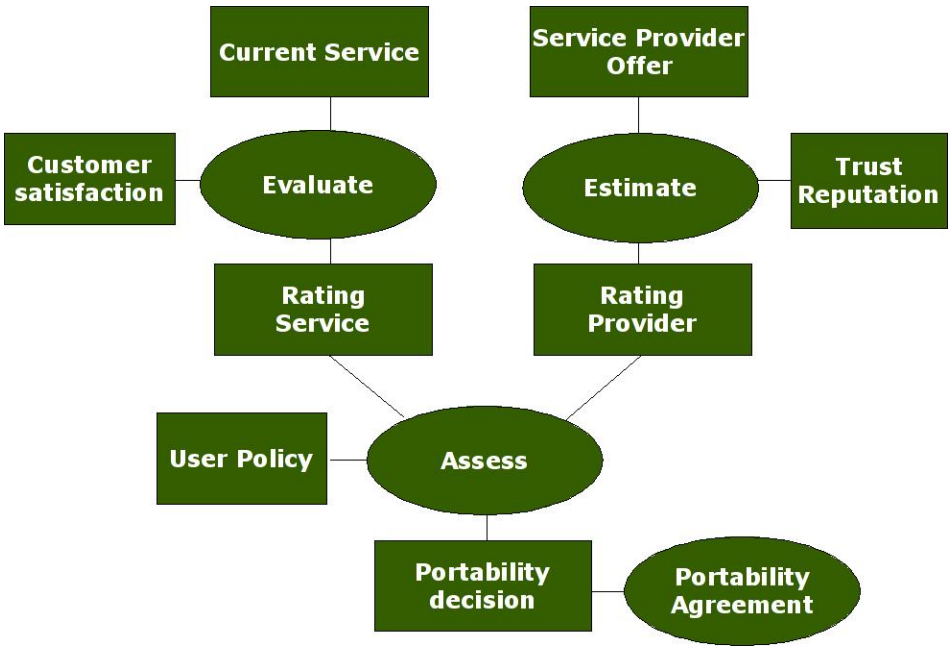


Fig. 4. Portability pattern reasoning diagram

Forces This pattern is applicable when there are more than one stakeholder in the development of the new service, and the consumer do not have strong penalties or commitments with the current provider.

Solution Based on the QoS evaluation of the current service provider, the estimation of the QoS of the available service providers, and the consumer policies and commitments, the decision to change of provider is taken, which involves requesting portability (breaking the current agreement with the service provider and establishing an agreement with the new provider).

Examples This pattern is frequently found in several domains. For example, it is a common pattern in telephony customers or mobile network environments.

Resulting context The consequences of the portability can be penalties due to commitments with the initial service provider, and the new commitment with the new service provider.

Duration Short term (specific project)

Topic Service Provision

Phase Agreement conclusion, Provider selection, Agreement negotiation

Decision making Agreement change

Related Patterns –

Known uses Wireless networks [35], professional services [36]

The context and resulting context of this pattern can be easily formalized as preconditions and postconditions as follows. Let be d a decisor, p_c the current

provider, p_i the available provider i, and the relation *provider(decisor, provider)* used for representing that *decisor* has an agreement in effect with the *provider*.

Let be $\mathcal{U}_i^d$ the utility function of a decisor d for a provider i.

In order to be applied this pattern, the following pre and postconditions should be fulfilled.

$$\mathfrak{Preconditions} \begin{cases} provider(d,c) \quad \wedge \\ \neg provider(d,i)\ \forall i, \quad i \neq c \quad \wedge \\ \exists i, \quad \mathcal{U}_i^d \geq \mathcal{U}_c^d \quad | \quad i \neq c \end{cases}$$

$$\mathfrak{Postconditions} \quad provider(d,i) \wedge \neg provider(d,c)$$

4.3 Intermediary Pattern

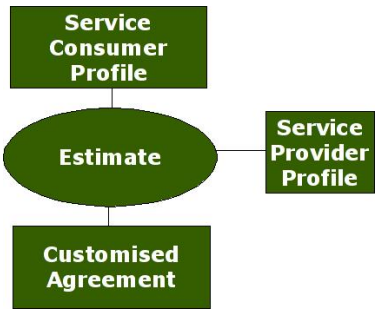


Fig. 5. Intermediary Pattern

Name *Intermediary*
Alias *Service Broker*
Duration Short term (specific project)
Topic Service Provision
Phase Agreement Negotiation
Problem Selection of a service provider that fits the QoS requirements of the consumer.
Context There are several available service providers that fit the functional requirements.
Forces Some of the considerations to be taken into account if the trust of the service consumer on the intermediary.
Solution The intermediary selects one or more providers and adapts the service provider offer to the service consumer preferences, providing an added value.
Examples Service provider selection in wireless networks [35], tourism and finance [37],

Resulting context the intermediary acts on behalf of the user based on an agreement between user and intermediary. This pattern refines the *service mediator* entity defined within SOA-RM [9].

Related Patterns Portability. Portability could be a decision of the intermediary.

5 Conclusions and Future work

This article presents preliminary results of the research on defining a software methodology for *agreement technology*. The task is challenging, since *agreement technology* is an emerging area.

The research has been based on the following premises:

- *Applicability*. The results of agreement technologies should be applied by any software component which requires to fulfill an agreement. In particular, we have not restricted the applicability to the agent community.
- *Simplicity*. Agile methodologies have shown their potential for fitting in every day task. In particular, design patterns and refactoring have succeed in its adoption. The results of our approach try to be easy to adopt.
- *Reuse*. In order to provide technology transfer from agreement technologies to other areas, it is needed to organize and provide facilities for the comprehension of the research results. Patterns provide a suitable mechanism for this technology transfer and a principled way.

The main conclusions of our research is that *agreement patterns* can be a useful concept for providing a unifying vocabulary in agreement technology, and providing a catalogue of methods for its implementation.

In this article we have proposed a first structure and some examples of application with some agreement patterns we have identified. Nevertheless, these are initial results, and we are now working on a more definitive structure of the patterns, as long as its formalization and operationalization.

Acknowledgements

This research has been partly funded by the Spanish Ministry of Science and Innovation through the projects Ingenio Consolider2010 AT *Agreement Technologies* (CSD2007-0022)) and T2C2 (TIN2008-06739-C04-03/TSI) as well as the Spanish Ministry of Industry, Tourism and Trade through the project RESULTA (TSI-020301-2009-31).

We would like to express our gratitude to Sascha Ossowski and Alberto Fernández for involving us in the project AT as well as the rest of the URJC research group for their support within the project. We would like also to thank the rest of research groups of AT, IIIA CSIC and UPV for their comments, and specially to Vicente Botti and Vicente Julián for their support, motivation and patience along the project.

References

1. Working Group on Future Internet Infrastructure FP7 ICT Advisory Group: Future internet infrastructure. Technical Report version 8, FP7 ICT Advisory Group (January 2008)
2. Sierra, C., Botti, V., Ossowski, S.: Agreement technologies. Available at http://www.agreement-technologies.org (2008)
3. Jennings, N.R., Faratin, P., Norman, T.J., O'Brien, P., Wiegand, M.E., Voudouris, C., Alty, J.L., Miah, T., Mamdani, E.H.: ADEPT: Managing business processes using intelligent agents. In: Proceedings of the 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems (ISIP Track), Cambridge, UK (1996) 5–23
4. Huhns, M.N., Singh, M.P., Burstein, M., Decker, K., Durfee, E., Finin, T., Gasser, L., Goradia, H., Jennings, N., Lakkaraju, K., Nakashima, H., Parunak, V., Rosenschein, J.S., Ruvinsky, A., Sukthankar, G., Swarup, S., Sycara, K., Tambe, M., Wagner, T., Zavala, L.: Research directions for service-oriented multiagent systems. IEEE Internet Computing **9**(6) (2005) 65–70
5. Lomuscio, A.R., Wooldridge, M., Jennings, N.R.: A classification scheme for negotiation in electronic commerce. Lecture Notes in Computer Science **1991** (2001)
6. Greco, G., Scarcello, F.: On the complexity of computing peer agreements for consistent query answering in peer-to-peer data integration systems. In: CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management, New York, NY, USA, ACM (2005) 36–43
7. Papazoglou, M.P.: Service -oriented computing: Concepts, characteristics and directions. Web Information Systems Engineering, International Conference on **0** (2003) 3
8. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web services agreement specification (WS-Agreement). Technical report, Grid Resource Allocation Agreement Protocol (GRAAP) Working Group (2007)
9. McCabe, F.G.: Reference architecture for service oriented architecture. Technical report, OASIS (April 2008)
10. Parashar, M., Hariri, S.: Autonomic Computing. CRC Press, Inc., Boca Raton, FL, USA (2006)
11. Carnevale, P.: Psychology of Agreement. Psychology Press (2009)
12. Jennings, N.: Agreement technologies. Intelligent Agent Technology, IEEE / WIC / ACM International Conference on **0** (2005) 17
13. Schmidt, D.C.: Using design patterns to develop reusable object-oriented communication software. Communications of the ACM **38**(10) (1995) 65–74
14. Rosengard, J.M., Ursu, M.F., marc Rosengard, J., Ursu, M.F.: Ontological representations of software patterns. In: Proc. KES'04, Lecture Notes in Computer Science, Springer Verlag (2004) 31–38
15. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: Pattern-oriented software architecture: a system of patterns. John Wiley & Sons, Inc. (1996)
16. Oluyomi, A., Karunasekera, S., Sterling, L.: A comprehensive view of agent-oriented patterns. Autonomous Agents and Multi-Agent Systems **15**(3) (2007) 337–377
17. Oluyomi, A.O.: Patterns and Protocols for Agent-Oriented Software Development. PhD thesis, Faculty of Engineering. University of Melbourne, Australia. (November 01 2006)

18. Erl, T.: SOA Design Patterns. Prentice-Hall (2008)
19. Rotem Gal Oz, A.: SOA Patterns. Manning (2009)
20. Zdun, U., Hentrich, C., Aalst, W.M.P.V.D.: A survey of patterns for service oriented architectures. Int. J. Internet Protoc. Technol. **1**(3) (2006) 132–143
21. Pěchouček, M., Mařík, Bárta, J.: A knowledge-based approach to coalition formation. IEEE Intelligent Systems **17**(3) (2002) 17–25
22. Camarinha-Matos, L.M., Afsarmanesh, H.: Collaborative networks. value creation in a knowledge society. In: Proceedings of PROLAMAT'06, Shanghai, China (2007)
23. Smith, R.G.: The contract net protocol: High-level communication and control in a distributed problem solver. Transactions on Computers **C-29**(12) (1980) 1104–1113
24. Ossowski, S.: Coordination in multiagent systems – towards a technology of agreement. In: MATES-2008. Volume 5244., Springer Verlag, Springer Verlag (2008) 2–12
25. Mui, L.: Computational Models of Trust and Reputation: Agents, Evolutionary Games, and Social Networks. PhD thesis, MIT (2002)
26. Kersten, G.: Support for group decisions and negotiations. an overview. In Climaco, J., ed.: Multicriteria Analysis, Springer Verlag (1997) 332–346
27. Schreiber, G., Akkermans, H., Anjewierden, A., Dehoog, R., Shadbolt, N., Vandevelde, W., Wielinga, B.: Knowledge Engineering and Management: The CommonKADS Methodology. The MIT Press (December 1999)
28. Maximilien, E.M., Singh, M.P.: A framework and ontology for dynamic web services selection. IEEE Internet Computing **8**(5) (2004) 84–93
29. Şensoy, M.: A Flexible Approach For Context-Aware Service Selection In Agent-Mediated E-Commerce. PhD thesis, Boğaziçi University (2008)
30. Şensoy, M., Yolum, P.: On choosing an efficient service selection mechanism in dynamic environments. In: Proceedings of the 9th International Workshop on Agent-Mediated Electronic Commerce (AMEC IX), 2007. Volume 13. (2007) 105–118
31. Şensoy, M., Yolum, P.: A comparative study of reasoning techniques for service selection. In: Proceedings of the 5th International Workshop on Agents and Peer-to-Peer Computing (AP2PC'06). Volume 123–134. (2006) 91–102
32. Billhardt, H., Hermoso, R., Ossowski, S., Centeno, R.: Trust-based service provider selection in open environments. In: SAC '07: Proceedings of the 2007 ACM symposium on Applied computing, New York, NY, USA, ACM (2007) 1375–1380
33. Sierra, C., Faratin, P., Jennings, N.R.: A service-oriented negotiation model between autonomous agents, Springer-Verlag (1997) 17–35
34. Keaveney, S.: Customer switching behaviour in service industries: an exploratory study. Journal of Marketing **59** (1995) 71–82
35. Lee, E.G., Lee, G., Faratin, P., Bauer, S., Wroclawski, J.: Automatic service selection in dynamic wireless network. Technical report, First ACM International Workshop WMASH'03 (2003)
36. Kugyté, R., Sliburyte, L.: A standardized model of service provider selection criteria for different service types: a consumer oriented approach. Engineering Economics. Commerce of Engineering Decisions **43**(3) (2005) 56–63
37. Yu, C.C.: A web-based consumer-oriented intelligent decision support system for personalized e-services. In: ICEC '04: Proceedings of the 6th international conference on Electronic commerce, New York, NY, USA, ACM (2004) 429–437

Achieving Mediated Agreements using Agreement Space Modeling

C. Carrascosa, M. Rebollo

Universidad Politécnica de Valencia
Departamento de Sistemas Informáticos y Computación (DSIC)
Camino de Vera s/n – 46022 Valencia – Spain
{carrasco, mrebollo}@dsic.upv.es

Abstract. An agreement is an arrangement between two or more entities to do something. This implies a context in which terms the agreement is developed. This paper presents a model of such agreement context as an euclidean space. On the other hand, an agreement can also be seen as a Constraint Satisfaction Problem (CSP) which solution space models an agreement space. This will allow to have a mediator that could not only model the agreement as an space, but also to check the viability of the agreement as it is being built. This mediator, that could even give some counsels about the agreement terms being formulated, is called here Counselor Agent. The protocols to interact with such agent to build an agreement are also presented.

1 Introduction

According to the Merrian-Webster dictionary, *agreement* is "an arrangement as to a course of action. Compact, treaty a: a contract duly executed and legally binding b: the language or instrument embodying such a contract". This agreement, this contract, assumes or implies a context in which it is going to be carried out. So, using a classical example by Castelfranchi [3] regarding the way an speech act tries to induce different motives for goal adoption in the addressee, when a general commands to a soldier 'Fire!', the soldier could interpret such an order in different ways according to the several meanings of such expression (he could shoot, or offer to the general some matches, or look for the firemen, or burn a fire, ...), but the most probable outcome is that he wouldn't doubt between such options and shoot his weapon, due to the context of the situation, that is, there is an implicit agreement between general and soldier regarding orders and military concepts.

This paper continues the work of [2][1] that define an agreement by means of a multi-dimensional Euclidean space approach. This modeling will allow to use geometrical properties or operations to work with such agreements, and even to model the agreement problem as a constraint-satisfaction problem (CSP) and to solve it using the Hyperpolyhedron Search Algorithm ($HSA \neq$) [11]. This paper focuses in the way to use this modeling as a CSP by a mediator (Counselor Agent), which could be able to on-line evaluate the feasibility of such

an agreement, or even give some advises as the agreement negotiation evolves. In order to do that, several agreement-related interaction protocols have been defined to be used by the participant agents to reach an agreement upon a set of agreed terms.

So, one of the main differences between this approach and previous works related with agreements and agents, is the focus of the approach. In works as *Contract-related agents* [8], the focus is in the agent, and the way he may deal with the concept of contract. The work here presented focuses in the concept of agreement as an upper abstraction over the idea of agent, organization or other entity. Modeling the agreement in this way, is what can be really useful to a counselor mediating in such agreement.

The rest of the paper is structured as follows. Context Spaces are introduced in section 2. Section 3 provides a set of general definitions needed to describe the proposed protocols and algorithms. Agreement-related interaction protocols are described in section 4 and section 5 explains how the Counselor Agents works to help participant agents to reach an agreement. A simple example is shown in section 6 and, finally, section 7 summarizes this approach and concludes.

2 Context Space

This term is used in the pervasive computing field related to context-aware computing [10] [9] to define a theory that models contexts based on intuitions from state-space models. In this way, the context space of an application is determined by the types of information, deemed relevant and obtainable by the designers, that is the context attributes, and the domain of possible values in each attribute. Each one of these attributes is considered one dimension in a multidimensional Euclidean space. This multi-dimensional space defines the space in which context can be perceived.

This approach also includes a context algebra to express situations in terms of Context Spaces and to reason about such situations by means of a set of operators and calculations. The operators defined are the *scalar difference* (it calculates the degree of similarity between two comparable context states), and the *intersection* (it produces a new context space containing shared regions of values of the same attributes between two comparable context spaces).

By representing situations and system's state as multi-dimensional objects, it is possible to generically describe and consequently reason about context of a system. A region of acceptable values is defined as a set which satisfies some predicate, hence, it can consist of any information (numerical or non-numerical) that best reflect the context attribute behavior (in terms of possible values) for the specific situation.

3 Agreements Concepts

This section presents some generic concepts and definitions related to agreements needed to explain the algorithms and protocols designed for a counselor agent to guide the agreement process. A more detailed definition can be found in [2][1].

3.1 Agreement Definitions

An *agreement* is the definition of a working context for two or more beings or entities (agents, organizations, ...) so that it is obtained as the result of a negotiation process.

Definition 1 (Agreement). *An agreement Ag is defined as $Ag = (E, Cx)$, where:*

- $E = \{E_1, E_2, \dots E_n\}$ *is a set of entities participating in* Ag*, so that* $\forall E_i, i = 1, \dots n, \exists! O_i = \{o_{ij}\}, j = 1, \dots m$*, ontology or set of concepts known by* E_i
- $Cx = \{(cx_o, cx_o^I) | cx_o \in \bigcup_i O_i, cx_o^I \subseteq D_o\}$*, where* D_o *is the domain of the ontology term* cx_o*. Thus, this context is formed by a set of ontological terms* cx_o *with its corresponding set of valid instances* cx_o^I *that have been agreed by* E*.*

Definition 2 (Agreement Discourse Universe). *The Agreement Discourse Universe of an agreement Ag –$ADU(Ag)$– is the whole set of concepts known by all the entities participating in the agreement. So, if O_i is the set of all the ontologies known by the entity E_i participating in the agreement Ag ($i = 1..n$), then $D = \{o / o \in O_i \triangle O_k, \forall i \neq k / i, k = 1..n\}$ (the symmetric difference of all the agreement participating entities' ontologies),*[1] *and the $ADU(Ag)$ is defined as $ADU(Ag) = \bigcup_i O_i - D$.*

The $ADU(Ag)$ is formed by all the concepts that at least one pair of entities participating in the agreement Ag share (Figure 1). If this set is empty, entities have not anything in common and they can not reach any kind of agreement.

3.2 Agreement Process

An agreement Ag has two phases:

1. Reaching an agreement (defining the agreement context Cx): It comprises the negotiation process between two or more entities (belonging to E) to reach an agreement. In fact, it is decided in two levels:
   (a) A decision must be taken about what are the concepts around which such agreement is going to be related, that is, $Cx \subseteq ADU(Ag)$.

[1] The symmetric difference is defined as $A \triangle B = (A \cap B') \cup (A' \cap B)$, where A' denotes the complementary set of A

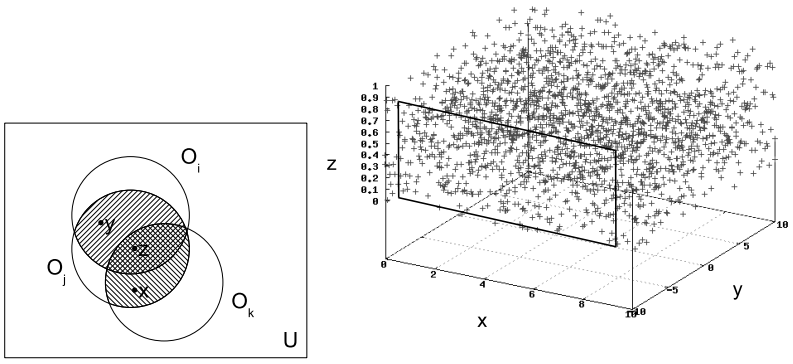


Fig. 1. Agreement Discourse Universe and Agreement Discourse Space with the projection to a 2-dimensional Agreement Space

(b) After that, the specific terms of such agreement must be fixed, that is, the values or intervals for the concepts in Cx.

2. Agreement execution: In this phase each entity must fulfill the accomplished agreement executing the needed actions or calculus according to the context defined by such agreement. This execution could not even imply any kind of additional coordination.

3.3 Agreement Space

As it has been stated before, an agreement can be seen as the definition of a common context. As commented in section 2 an space metaphor such as the one used in Context Spaces can be used to express such contexts. So, an *agreement space* can be defined.

Definition 3 (Agreement Discourse Space). *The Agreement Discourse Space of an agreement* Ag $ADS(Ag)$ *is defined by considering as a dimension (in an Euclidean space) each concept included in an Agreement Discourse Universe of an agreement Ag. That is, the $ADS(Ag)$ is an n-dimensional space, where n is the cardinality of the $ADU(Ag)$ (number of common terms). Figure 1 shows an example with 3 dimensions, where the points represent the possible combination values of such space (assuming a discrete domain for every dimension).*

Definition 4 (Agreement Space). *According to the first phase of an agreement, a decision must be taken about the concepts, that is, the dimensions in which the agreement will be expressed. So, an Agreement Space is a projection of the Agreement Discourse Space onto the dimensions defining the agreement. That is, this space will be defined by the features the different entities E_i making the agreement are going to negotiate (Cx), each one of such features defining a dimension in this space ($\forall i : d_i \in dim(E_i, Ag)$).*

In order to be possible an agreement Ag, for all entity E_i participating in the agreement there will exist at least one other participating entity E_j so that $dim(E_i, Ag) \cap dim(E_j, Ag) \neq 0$.

The Agreement Space is, then, the result of eliminating the unnecessary dimensions that are not relevant for the agreement. For example, in Figure 1 the y dimension is not relevant, so the way the entities will deal with y concept is not controlled by the agreement, and the Agreement Space is reduced to the marked 2-dimensional area, that is, the projection over x and z dimensions of the defined ADS.

In this way, the outcome of the first phase of the agreement will be the definition of this Agreement Space, fixing the satisfying values for each one of the different dimensions comprising this space. The second phase of the agreement will be to carry out the execution of the agreement taking into account that it has to be carried out inside the previously defined Agreement Space.

Definition 5 (Local Agreement Space). *The* Local Agreement Space *for entity E_i in the agreement Ag is defined as the projection over the dimensions of interest of entity E_i in such Agreement Space.*

4 Agreement-Related Interaction Protocols

WS-Agreement[6] is a standard proposed by the Global Grid Forum (GGF) that includes the definition of a simple interaction protocol to support one-to-one negotiation, with the likely aim to support different mechanisms in the future through definition of multiple interaction protocols. So, the work by [7] for what they called *service agreement* where one consumer of a service chooses one service provider from n available by means of the FIPA Iterated Contract Net Interaction Protocol[5].

In this proposal, a Counselor Agent (see Section 5) mediates in the agreement process in order to help the participant agents to reach an agreement related to a common vocabulary. The mediation is achieved though a set of services available to all participants that help to find a common Agreement Space, inside which agents can interact being ensured that the terms of the agreement are fulfilled.

This section presents two new interaction protocols that has been defined to deal with the services offered by the Counselor Agent: *ADU Definition* and *Agreement Counselor*, that will be addressed by the *ADU Interaction Protocol* and the *Mediated-Agreement Interaction Protocol*, respectively.

4.1 ADU Interaction Protocol (ADUIP)

This interaction protocol controls the service of the Counselor Agent related to the definition of the *Agreement Discourse Universe* of a group of entities, that is, the common vocabulary shared by such entities (see Definition 2). This protocol describes a general phase, previous to the agreement itself, that can be used in many other situations in which establish a common vocabulary is needed. Figure 2 shows this protocol, that is composed of the following steps:

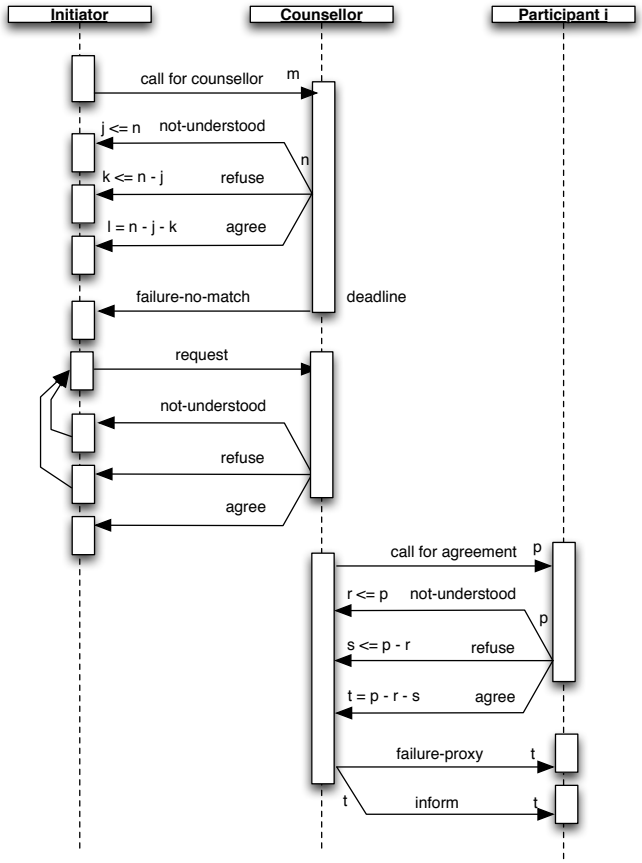


Fig. 2. ADU Interaction Protocol

1. An Entity E_i sends a *Call For Counselor* to all entities offering the service of *Counselor* indicating an $AgreementID$.
2. If any Counselor agrees to E_i, then E_i will send a *Request* for Counselor to one of them, C_j, that will serve as counselor for agreement $AgreementID$.
3. Counselor C_j sends a *Call For Agreement* to all entities in the system, indicating an *Agreement ID* and a deadline to this phase of joining to the agreement.
4. Each entity E_k that wants to participate in the agreement will send to C_j an agree message with his ontology before deadline is met.
5. After deadline, C_j will calculate the ADU using the ontologies received from the entities that will participate in the agreement, communicating them if the ADU is empty that there is a failure, because there is no possibility of agreement.

The ADU Interaction Protocol is only focused on provide a framework to participant agents for exchanging their ontologies (or fragments of them). It does not try to solve the ontology alignment problem, which is out of the scope of this paper.

4.2 Mediated-Agreement Interaction Protocol

After the existence of a common vocabulary is ensured by ADU protocol, the participant entities can negotiate the terms of any agreement. A two-phases protocol is proposed to reach an agreement among a group of entities (agents or organizations) mediated by a counselor. The final goal of this protocol is to construct the Agreement Space by means of the following steps:

1. Defining the context: It will be formed by the dimensions of the Agreement Space, that is, the concepts that will be dealt in the agreement, and that will be part of the ADU (in fact, this context will be a projection of the ADU in the dimensions of the Agreement Space).
2. Defining the agreement terms, that is the different values or intervals for the dimensions defining the context that will define the Agreement Space. This process will be done by applying one by one the constraints suggested by the participants. Moreover, it is regulated by a mediator—the Counselor—which guides the negotiation process to achieve an accord among the participants. Its main task is to check that any additional constraint is consistent with the current constraint set. That is, if they formed a convex hull.

The protocol here presented is described from the Counselor point of view, and, its purpose is to help in the Agreement Space definition. This protocol (see Figure 3a) is divided into two main parts: the first one deals with the definition of the context, and the second one with the definition of the agreement terms (see Figure 3b).

1. Context Definition: An initiator requests the Counselor for an agreement.
   (a) An Entity E_i sends a *Call For Agreement* with $AgreementID$ to the Counselor Agent that has previously calculated the ADU, indicating that he wants to begin an agreement with some or all the other entities involved in such ADU.
   (b) If the Counselor agrees to E_i, then he sends a *Call For Context* to all the entities associated to the previous ADU, indicating an *Agreement ID* and a deadline to this phase of joining to the agreement.
   (c) Each entity E_k that wants to participate in the agreement will send to the Counselor an *Agree* message with his agreement dimensions (that is, the concepts he is interested in negotiate for the agreement, and that are part of the previous ADU) before deadline is met.
2. Agreement Terms Definition: The former initiator is turned into another participant and the Counselor takes the initiative.

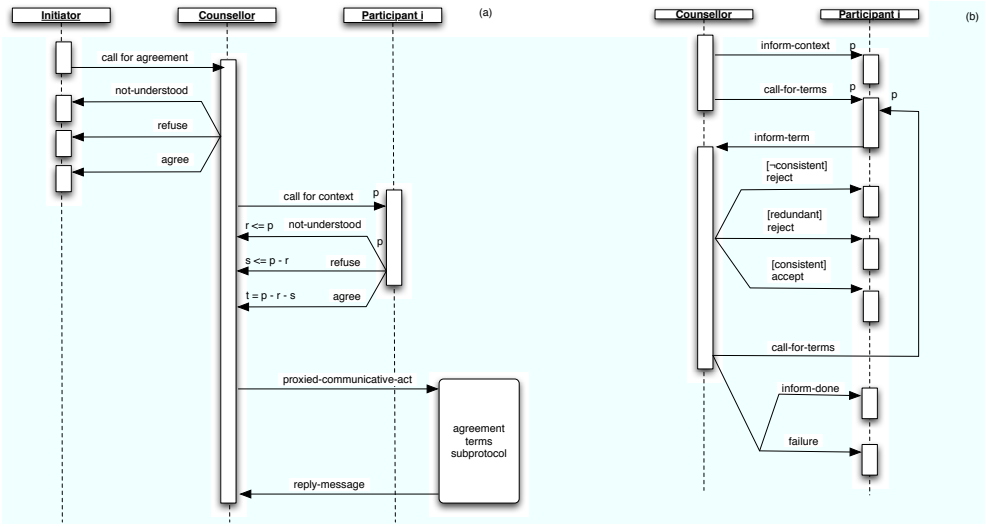


Fig. 3. (a) Mediated-Agreement Interaction Protocol (b) Agreement terms definition subprotocol

(a) The Counselor *Informs* to all the entities of the dimensions of the agreement (from the general ADU, different concrete agreements could take place, between different or the same entities).

(b) The Counselor sends a *Call For Terms* message to all the participating entities with the proper $AgreementID$ to ask for agreement terms.

(c) Each one of the participating entities will send an *Inform* message to the Counselor for each one of the negotiation terms (in the form of a constraint) that he is interested in. Such messages will be processed by the Counselor to build the AS.

(d) For each *Inform* message, the Counselor will respond either with a *Reject* message, if the new term is not consistent with the agreement or is redundant, or with an *Accept* message if the new term is consistent.

(e) The process of receiving and analyzing new agreement terms continue till there is no more terms (according to a deadline-based method).

(f) The Counselor informs the participating entities either the final agreement terms, or that the agreement has not been possible.

The final result is the definition of the Agreement Space, modeled as a hyper-polyhedron formed by all accepted constraints (agreement terms). During the execution of the agreement, all interactions have to be inside this space as the participants have agreed.

5 Using Agreement Spaces for Mediators

It is not so strange, when trying to reach an agreement to use a mediator to, not only watch over the fulfillment of the different processes involved in reaching an agreement, but also to counsel about the final values or intervals of the agreement due to its global unbiased view of the agreement. So, this figure is also interesting when trying to automatize the agreements, that is, to carry out agreements between agents, virtual organizations or other pieces of software. Different approaches to automatic mediators can be seen in the literature such as the PERSUADER system [12] or AutoMed [4].

5.1 Agreement Space as a CSP

As it was stated before, after the Agreement Space is defined, the different entities involved in the agreement must fix (possibly through a negotiation process) some concrete values or intervals for the different dimensions comprising the Agreement Space. For such negotiation, each one of these entities will have some intervals of satisfying values for the different dimensions it is interested in. These satisfying intervals may be expressed as constraints for the different dimensions that should be satisfied for the agreement. So, if this first phase of an agreement is seen from a mediator point-of-view, it can be seen as the solution of a Constraint-Satisfaction Problem (CSP), defined by the whole set of constraints of the different entities. Being the Agreement Space n-dimensional, each one of such constraints defines a $(n-1)$-dimensional plane. As the Agreement Space is defined by the intersection of all such planes, it can be described how such space is by studying such planes. Therefore, this *agreement mediator* should be able to detect if an agreement is possible, checking if there exists an Agreement Space as a result of all these constraint planes intersection.

The idea is to define the Agreement Space as the solution space of a CSP and to build an agreement by applying entities' conditions as constraints in the space (applying the $HSA \neq$ [11] algorithm to build the hyperpolyhedron modeling the CSP that will define the Agreement Space). A solution is reached if all constraints are consistent. Furthermore, if the resulting space is convex then it can be ensured that all interactions are inside the Agreement Space.

As it is an agreement, an Agreement Space (AS) is defined by a set of entities and their context (E, Cx), where the context is composed by variables, their domains an all the constraints. In this way, the AS is defined by extending a common CSP definition.

5.2 Counselor Algorithm

The Counselor has the responsibility of helping the participants in the agreement to find the Agreement Space, considering it as a CSP. It is an iterative process that adds new constraints one by one, guaranteeing that the new constraint is consistent with the previous ones. Constraints are inequalities of the form $\sum_{i=0}^{n} a_i x_i \leq b$, where x_i are the variables of the CSP. The set of all constraints

defines an hyperplane and which union defines the limits for the valid values for all the variables involved in the agreement. The participants in the agreement will take values inside the resulting hyperpolyhedron.

Using the mediated-agreement protocol defined in Section 4.2, the Counselor obtains the constraints from the participants and creates the Agreement Space using a variant of $HSA\neq$ algorithm. This algorithm is applied inside the Agreement Terms Definition subprotocol (see Figure 3b) each time a new constraint C_i is received from a participant (by means of an *Inform* message). So the Agreement Space is incrementally constructed from scratch.

If C_i is inconsistent or redundant, the agent that proposed it is informed by a *Reject* message. The agent can redefine the constraint and submit a new one if necessary.

Consistence and redundancy are checked using the $HSA\neq$ algorithm. Basically, when a new constraint C_i arrives the algorithm checks if each vertex of the current hyperpolyhedron satisfies C_i. If none vertex satisfy C_i then the constraint is inconsistent and it is rejected. That is because the space defined by the constraint does not contain any of the vertex of the hyperpolyhedron. On the other hand, if all vertex satisfy C_i then the constraint is redundant. That is because the complete hyperpolyhedron is included inside the space defined by the constraint. In any other case, the constraint is consistent and it is added to the hyperpolyhedron.

When all constraints are added to the Agreement Space and it is not empty then there is an agreement among all the participants. If the Agreement Space is not empty, then it is ensured that it is convex, so any 'movement' between two points inside this space is always inside it. Therefore, it is guaranteed that any negotiation process produced inside the agreement terms will never violate the agreement.

The following section presents an example of reaching an agreement using a Counselor Agent, interacting with the above presented interaction protocols, focusing in the Agreement-terms definition subprotocol.

6 Example

Let it be a set of agents interested in taking piano classes. The group is formed by one teacher and two students. They agree to negotiate over three dimensions: the number of classes (n), its duration (d) and its price (p). Initially, participants have their own preferences (modeled as constraints).

- Teacher
  1. at least 10 classes: $n \geq 10$
  2. duration between 60 and 120 min: $d \geq 60$ and $d \leq 120$
  3. at least 20 euros/hour: $p \geq 20$
- Student 1
  1. no more than 20 classes: $n \leq 20$
  2. less than 30 euros/hour: $p \leq 30$

– Student 2
  1. minimum 15 classes: $n \geq 15$
  2. duration between 45 and 90 min: $d \geq 45$ and $d \leq 90$

Assuming that agents have some common vocabulary (checked with the ADU protocol) and they defined the above three dimensions as the agreement context, then the Counselor asks them for the terms of the agreement. There is no rule about the order in which agents propose the terms (constraints) nor the order in which they arrive to the Counselor. Depending on it, one constraint can be detected as redundant or just as a refinement of an existing one. But it does not affect to the final Agreement Space.

	Agent	Mess	Param		Agent	Mess	Param		Agent	Mess	Param
1	C	inform	n, d ,p	11	C	call-for-terms		21	T	inform	$d \leq 120$
2	C	call-for-terms		12	T	inform	$d \geq 60$	22	C	accept	$d \leq 120$
3	T	inform	$n \geq 10$	13	C	accept	$d \geq 60$	23	C	call-for-terms	
4	C	accept	$n \geq 10$	14	C	call-for-terms		24	T	inform	$p \geq 20$
5	C	call-for-terms		15	S1	inform	$n \leq 20$	25	C	accept	$p \geq 20$
6	S1	inform	$p \leq 30$	16	C	accept	$n \leq 20$	26	C	call-for-terms	
7	C	accept	$p \leq 30$	17	C	call-for-terms		27	S2	inform	$d \leq 90$
8	C	call-for-terms		18	S2	inform	$d \geq 45$	28	C	accept	$d \leq 90$
9	S2	inform	$n \geq 15$	19	C	reject	$d \geq 45$, red	29	C	inform-done	
10	C	accept	$n \geq 15$	20	C	call-for-terms					

Table 1. Example of Agreement Terms Definition

This is a very simple case, in which restriction are defined over one dimension, but it is enough to follow and understand the process of construction of the Agreement Space. Let be C the counselor, T teacher, S1 student 1 and S2 student 2. An example of the process can be followed in Table 1. In the sequence of messages on this table it can be seen how the Counselor communicates to the rest of participants the context of the agreement (message 1). After that, the participants send their own constraints one by one (messages 3, 6, 9, 12, 15, 18, 21, 24 and 27) and the Counselor accepts or rejects them. For example, in message 18, S2 proposes a constraint that is redundant, since one more restrictive has been accepted in 12-13. But restrictions proposed in messages 15 and 27 have been accepted because they refine (they are more restrictive) that the ones accepted in 6-7 and 21-22 respectively.

7 Conclusions

Context Spaces seem to be a valid approach to model agreements and allows to use well known techniques to solve the task of reaching an agreement and controlling its subsequent execution. The agreement is modeled as an object in the Euclidean space.

The problem of defining the space associated to an agreement can be seen as a CSP, since the agreements conditions can be modeled as constraints and the

agreement is expressed as a set of restrictions over a set of data. The conjunction of all these restrictions will define an convex hyperpolyhedron delimiting the *agreement space*, that is, the space where all the computations fulfilling such agreement will be carried out, the *context* of such computations.

Mediators can be used during the negotiation process to reach an agreement to check that any additional constraint keeps this space convex, along with controlling the feasibility of the agreement. For doing that, mediators can suggest more relaxed or more strict constraints in order to maintain the consistence of the space and the possibility of an agreement.

This paper has presented not only the advantages of using such a Mediator or Counselor in reaching an agreement, but also the way this Counselor works and how to interact with him by means of some interaction protocols specifically designed for agreement reaching processes.

As future work, the dynamic of the agreement is going to be studied, because to guarantee an agreement, besides the existence of the agreement space, the convergence of individual negotiation processes has to be guaranteed to avoid systems that oscillate without ending in a concrete agreement for some initial situations.

References

1. C. Carrascosa and M. Rebollo. Modelling agreement spaces. In *IBERAMIA 2008 Workshop on Agreement Technologies (WAT 2008)*, pages 79–88. Marc Esteve, Adriana Giret, Alberto Fernández, Vicente Julián, 2008.
2. C. Carrascosa and M. Rebollo. Agreement spaces for counselor agents (short paper). In *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*. Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10-15, 2009, Budapest, Hungary, To appear, 2009.
3. C. Castelfranchi. Prescribed mental attitudes in goal-adoption and norm-adoption. *Artificial Intelligence and Law*, 7–1:37–50, 1996.
4. M. Chalamish and S. Kraus. Automed: an automated mediator for bilateral negotiations under time constraints. In *AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–3, New York, NY, USA, 2007. ACM.
5. FIPA. *FIPA Iterated Contract Net Interaction Protocol Specification*. FIPA, 2001.
6. G. Forum. *Web services agreement specification (ws-agreement)*. G. G. Forum, 2004.
7. F. Ishikawa, N. Yoshioka, and S. Honiden. Incorporating agreements on service options into bpel-based services. In *CIMCA '05: Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-1 (CIMCA-IAWTIC'06)*, pages 796–803, Washington, DC, USA, 2005. IEEE Computer Society.
8. J. Knottenbelt. *Contract Related Agents*. PhD thesis, Department of Computing, Imperial College London, December 2006.
9. A. Padovitz, C. Bartolini, A. Zaslavsky, and S. W. Loke. Extending the context space approach to management by business objectives. In *HP OpenView University*

Association (HP-OVUA), 12th Workshop, Porto, Portugal (FEUP), 10 - 13 July 2005, 2005.
10. A. Padovitz, S. W. Loke, and A. Zaslavsky. Towards a theory of context spaces. In *Workshop on Context Modeling and Reasoning (CoMoRea), in C Das and M Kumar (eds), Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshop, Orlando Florida, 14 - 17 March 2004*, pages 38–42, USA, 2004. IEEE Computer Society.
11. M. A. Salido, A. Giret, and F. Barber. Constraint satisfaction by means of dynamic polyhedra. In *Operational Research Proceedings 2001*, pages 405–412. Springer Verlag, 2001.
12. K. P. Sycara. Resolving goal conflicts via negotiation. In *Proceedings of AAAI-88*, pages 245–250, 1988.

Reputation-based Agreement for Agent Organisations*

Ramón Hermoso[1] Roberto Centeno[1] Viviane Torres da Silva[2]
{ramon.hermoso, roberto.centeno}@urjc.es and
viviane.silva@ic.uff.br

[1] Centre for Intelligent Information Technologies (CETINIA)
University Rey Juan Carlos Madrid (URJC) - Spain
[2] Universidade Federal Fluminense (UFF) - Brazil

Abstract. Reputation mechanisms have been proved to be valid methods to select partners in organisational environments. In order to tackle some well-known problems inherent to both centralised and distributed reputation mechanisms, a hybrid mechanism combining both techniques seems to be a promising approach. In this work firstly we summarise our previous work, a hybrid reputation mechanism, focusing on the distributed part. Then we put forward the centralised module that completes the mechanism, based on a novel concept such as *reputation-based agreement*, that attempts to define reputation aggregation as a global agreement reached amongst a set of participants belonging to an organisation. Besides, some particular properties of this type of agreements are proposed. The rest of the paper deals with the problem of how to present the information related to those agreements to agents in the organisations. For that, we will use *informative mechanisms* supported by some simple examples to better understand their functioning.

1 Introduction

Reputation mechanisms have been proved to be successful methods to build multi-agent systems where agents' decision-making processes to select partners are crucial for the system functioning [1][7][10]. In those systems, agents exchange their opinions about third parties to better select partners to interact with. In organisational environments, those mechanisms may also be useful for agents, since organisational structures, such as *roles* or *norms* can be used in order to estimate other participants' behaviour. Thus, integrating those organisational concepts into reputation mechanisms seems to be a promising approach to improve agents' decision making processes.

To cope with the interchange of opinions, many different reputation mechanisms have been proposed in the literature [7][10][13]. We can distinguish among

* The present work has been partially funded by the Spanish Ministry of Education and Science under project TIN2006-14630-C03-02 (FPI grants program) and by the Spanish project "Agreement Technologies" (CONSOLIDER CSD2007-0022, INGENIO 2010)

two different types, namely: *i)* centralised and *ii)* distributed. Both types have been proved to suffer from some important problems, such as: *newcomers* problems, lack of reliability in information exchanging, etc. [12]. To tackle with these problems, hybrid reputation mechanisms – combining centralised and distributed approaches – might be considered as a convenient technique [6]. In [12] we put forward a hybrid reputation model for organisations with a few strokes of the brush. We claimed, in that paper, that reputation fluctuates by the effect of norm fulfilment and violations, and that reputation values must be justified somehow by including, for instance, the norms that have been previously violated and the facts that have violated the norms. After that, in [5] we focus deeper on the distributed part, dealing with agents' information exchange, exploring the scenario of supply chains' formation. In the present work we present a formalisation of the centralised approach so completing the model.

In this paper we introduce the concept of *reputation-based agreement* as the cornerstone of the centralised module of the hybrid reputation mechanism. An *agreement* is usually defined as a meeting of minds between two or more parties, about their relative duties and rights regarding current or future performance. Around this concept new paradigms have been emerged [2][3] oriented to increase the reliability and performance of agents in organisations by introducing in such communities these well-known human social mechanisms. With this in mind, we propose a novel approach for the meaning of reputation. From a centralised point of view, a *reputation-based agreement* is a meeting point on the behaviour of an agent, participating within an organisation, with regard to its reputation. Agreements are evaluated by aggregating opinions sent by participants about the behaviour of the agents. We also define some interesting properties that describe different types of agreements. Information about reached agreements will be provided to agents by using the concept of informative mechanism [4].

The paper is organised as follows: Section 2 presents the previous work already done and in which the current paper is based on. Section 3 formalises the centralised module of our approach, supported on the idea of reputation-based agreements. In Section 4 we illustrate all concepts introduced by means of a case study. Section 5 discusses some related work and, finally, in Section 6 we summarise the paper and present the future work.

2 Previous Work

Reputation mechanisms are being used to increase the reliability and performance of virtual societies (or organisations) while providing mechanisms for exchanging reputation values. In centralised reputation models, a reputation system receives feedback about the interactions among the agents. Each agent evaluates the behaviour of the agents with whom it interacts and informs the reputation system. The system puts together all evaluations and stores such reputations. In contrast, in distributed reputation models, each agent evaluates and stores the reputations of the agents with whom it has interacted with and is able to provide such information to other agents.

With the aim to cope with the problems of centralised and distributed reputation mechanisms[3], we proposed the use of a hybrid mechanism [12]. In the distributed part of such a mechanism, agents evaluate the behaviour of other agents by exchanging opinions and storing such information. An opinion has to be justified by providing, for instance, the set of violated norms that contribute to that opinion.

This work is framed in organisational environments that provide a minimum set of organisational mechanisms to regulate agents' interactions. Formally, an organisation is defined as a tuple $\langle \mathcal{A}g, \mathcal{A}, \mathcal{X}, \phi, x_0, \varphi, \{\mathcal{ON}^{om}, \mathcal{R}^{om}\}\rangle$ where $\mathcal{A}g$ represents the set of agents participating within the organisation; $\mathcal{A}$ is the set of actions agents can perform; $\mathcal{X}$ stands for the environmental states space; ϕ is a function describing how the system evolves as a result of agents actions; x_0 represents the initial state of the system; φ is the agents' capability function describing the actions agents are able to perform in a given state of the environment; $\mathcal{ON}^{om}$ is an organisational mechanism based on organisational norms; and $\mathcal{R}^{om}$ is an organisational mechanism based on roles that defines the positions agents may enact in the organisation (see [5] for more details).

Agents participating in the field of such organisations are involved in different *situations*. A situation is defined as a tuple $\langle \mathcal{A}g, \mathcal{R}, \mathcal{A}, T\rangle$, that represents an agent $\mathcal{A}g$, playing the role $\mathcal{R}$, while performing the action $\mathcal{A}$, through a time period T. As detailed in [5], different types of situations can be defined following this definition. For instance, situations in which an agent performs an action, regardless of the role it is playing – $\langle \mathcal{A}g, \_, \mathcal{A}, \mathcal{T}\rangle$ –, or situations in which an agent is playing a role along a time period, regardless the action it performs – $\langle \mathcal{A}g, \mathcal{R}, \_, \mathcal{T}\rangle$ – are examples of possible situations.

As we aforementioned, we claim that when agents exchange opinions, those should be justified somehow, in order to allow receivers to reason about them (see [5] for more details) and, what is more important, this justification has to be based on the fulfilment of norms that regulate the different situation in which agents are involved. We consider two different types of norms regarding an organisation and its members. On the one hand, there exists norms that regulate all the participants in the organisation in different situations, known by all of them, which fulfilment could possibly be controlled by some authority entity. We call these norms *organisational norms*. Furthermore, we also define another type of norms – *personal norms* –, that regulate the preferences an agent has, regarding an individual situation. That is, they regulate how an agent wants a particular situation to be carried out. Agents define their own personal norms and they are the only ones that check their fulfilment. Note that, the personal norms defined by an agent regulates the behaviour of its partners and not its own behaviour, of course. As already pointed out, when an agent A sends an opinion to B about C – usually a reputation value –, A has to justify such value by sending the set of organisational norms that C violated when interacted with it, as well as the facts that prompted that reasoning. Moreover, personal norms that also contribute on an agent's reputation evaluation, are sent only when requested on

[3] In Section 5 we detail those problems

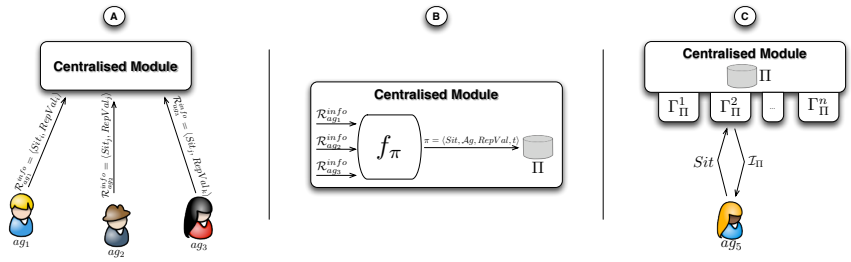


Fig. 1. Dynamics of the centralised module

demand. Starting from this approach, we focus on how to model the centralised part of the mechanism, stressing in the definition of *reputation-based agreement*.

3 Centralised Module based on Reputation-based Agreements

As we have previously pointed out, the current work faces with the task of formalising the centralised module to complete a hybrid reputation approach, working on organisational multiagent systems. The dynamics of such a module is threefold (as illustrated in Figure 1): *i)* agents within an organisation have to send their opinions about situations in which they have been involved; *ii)* the centralised module aggregates all opinions received from agents, creating *reputation-based agreements*; and *iii)* information about the agreements reached within the organisation is provided to agents by using different informative mechanisms [4]. In following sections we explain each task in detail.

3.1 How Agents Send Their Opinions

Along the lifetime of an agent within an organisation, it is involved in several different *situations* [5]. Usually, agents evaluate those situations in order to compile reliable information that allows them to predict the result of future situations. The rationale of the current work is that if agents share their knowledge about the situations they are involved in, this information might be useful when other agents have not enough information to select partners to interact with. This problem becomes hard when new participants join an organisation and they have not formed their own opinions so far.

The centralised module relies on the concept of *situation*. Situations are evaluated from an individual point of view. An evaluation may reflect the experience of the agent performing the evaluation – direct way – or the opinions provided by third parties about the evaluated situation – indirect way. Thus, at any time, an agent can send its opinion about a particular situation to the centralised module. Obviously, an agent can only send an opinion about a situation it has been

involved in: the centralised module could check this. We call this information *reputation information message* and it is formalised as follows:

Definition 1. *A reputation information message* $\mathcal{R}^{info}_{ag_i \in \mathcal{A}g}$ *is a tuple, representing an opinion sent by the agent* ag_i *to the centralised module containing an evaluation about a particular situation,* $\mathcal{R}^{info}_{ag_i} = \langle Sit, RepVal \rangle$,

where ag_i stands for the agent, which sends the opinion; Sit is the situation being evaluated; and $RepVal$ represents the evaluation the agent is sending about the situation (typically a number).

In this work we assume that agents are motivated to collaborate by sending their opinions to the centralised module. It is out of the scope of this paper to deal with the problem of lack of collaboration. In such situations, the centralised module should be coupled with an incentive mechanism that motivates agents to send their opinions. For instance, the module could give some credit to agents when they send an opinion, and later on, they could change that credit by information. Thus, an agent could be motivated to send their messages since it will be able to get useful information later on.

3.2 Creating Reputation-based Agreements

In this section we intend to face the task of giving a novel approach for the meaning of reputation, from a centralised point of view, tackling this concept as a partial agreement about a certain situation. When the centralised module receives reputation information messages from agents, it aggregates them creating what we have called *reputation-based agreements*. That is, the aggregation of all the opinions regarding to a particular situation is '*per se*' what that set of agents – as a whole – actually think about the aforesaid situation. Thus, a reputation-based agreement represents the consensus reached in the reputation opinions space sent by a set of agents about a particular situation. Formally:

Definition 2. *A reputation-based agreement* π *for a particular situation, is a tuple* $\langle Sit, \mathcal{A}g, RepVal, t \rangle$

where Sit stands for the situation about which the agreement is reached; $\mathcal{A}g$ is the set of agents that contributed to the agreement; $RepVal$ represents the reputation value – whatever its representation is (qualitative, quantitative, etc.) – reached as consequences of all opinions sent about the situation; and t stands for the time when the agreement was reached.

Therefore, an agreement means a global opinion that a set of agents have on a certain situation. This agreement, as we put forward in the next section, can be used as a generalist expectation for a situation in which agents have no (or little) previous information about.

As we have claimed, a reputation-based agreement is reached as consequence of the aggregation of all opinions sent about a particular situation. Thus, the centralised module requires a function that is able to aggregate information reputation messages sent by agents. The aim of such a function is to create

reputation-based agreements from reputation opinions that agents send to the module by means of reputation information messages. We formally define the function as follows:

Definition 3. *Let f_π be a function that given all the reputation information messages sent by agents and a particular situation creates a reputation-based agreement for that situation:*

$$f_\pi : |\mathcal{R}^{info}_{ag_i \in \mathcal{A}g}| \times Sit \rightarrow \Pi$$

where:

- $|\mathcal{R}^{info}_{ag_i \in \mathcal{A}g}|$ stands for the set of reputation information messages received by the centralised module;
- Sit is the set of situations;
- Π represents the set of reputation-based agreements.

As aggregation function the module might use any function that is able to aggregate values without any modification. For instance, it is possible to use a simple function to calculate the average of all opinions or a sophisticated function that aggregates the opinions by means of complex calculations (the implementation of this function is out of the scope of this paper). Note that a "good" aggregation function should take into account: i) the temporality of given opinions, since two opinions should not have the same importance if one of them was sent more recently; ii) in addition, the module should recalculate existing agreements when new opinions come.

3.3 Reputation-based Agreements: Properties

From previous definitions – definitions 2 and 3 – it is possible to define some desirable properties about reputation-based agreements. These properties should be taken into account when agreements are created and may also provide useful extra information when informing about different issues.

Property 1 *A reputation-based agreement π is **complete** iff. all agents participating in an organisation, at time t, contribute to reach that agreement:*

$$\pi^* \Leftrightarrow \begin{cases} \mathcal{O} = \langle \mathcal{A}g, \mathcal{A}, \mathcal{X}, \phi, x_0, \varphi, \{\mathcal{ON}^{om}, \mathcal{R}^{om}\}\rangle \ \wedge \\ \pi = \langle Sit, \mathcal{A}g', RepVal, t\rangle \ \wedge \\ (\mathcal{A}g = \mathcal{A}g') \end{cases}$$

That is, given a time t every participant $ag \in \mathcal{A}g$ in the organisation $\mathcal{O}$ has necessarily sent a reputation information message indicating its opinion about the situation concerning the agreement ($\mathcal{A}g = \mathcal{A}g'$). The more complete agreements are in the system, the more reliability the information will be offered.

Property 2 *A reputation-based agreement π is α-**consistent** iff. the reputation value of π differs, at most, α from the reputation value sent by every agent that contributed to reach that agreement:*

$$\pi^{\alpha} \Leftrightarrow \begin{cases} \pi = \langle Sit, \mathcal{A}g, RepVal, t \rangle \ \ \wedge \\ \forall ag \in \mathcal{A}g \ \ [\forall r \in Rep_{ag}^{info}[(r = \langle Sit_i, RepVal_i \rangle) \wedge \\ (Sit_i = Sit) \ \ \wedge \ \ (|RepVal_i - RepVal| \leq \alpha)]] \end{cases}$$

This property represents the monotony on the agents' behaviour, since measures how equals the opinions coming from them are. Therefore, the lower α is, the similar the opinions are.

Property 3 *A reputation-based agreement π is **full** iff it is complete and 0-consistent:* $\pi^{\phi} \Leftrightarrow (\pi^{*} \ \ \wedge \ \ \pi^{\alpha} \ \ \wedge \ \ \alpha = 0)$

In the case α is 0 means that all agents have the same opinion about a given situation. This property is very desirable when seeking reputation-based agreements, because the more agents contribute to the agreement, the stronger validity the latter has. Thus, the likelihood of capturing what is actually happening in the organisation tends to be higher.

Although properties 1 and 3 are desirable, they are not achievable in some types of systems, for example in electronic marketplaces. However, many systems can present those properties, such as closed organisational systems where the number of participants is not huge.

3.4 Providing Information about Reputation-based Agreements

Once we have defined an agreement as a distributed consensus-based expectation for a set of agents on a certain situation, we now present how the centralised model can present the relevant information on the reached agreements. Reputation-based agreements somehow capture the general thinking about a particular situation – when the less α-consistent the agreement is the more the reality captured is. Thus, information about the agreements reached until that moment may be very useful for agents. In particular, when agents have recently joined the organisation, they do not have any clue about situations in which they might be involved in, so if the centralised module provides information about agreements, agents may improve their utility from the very beginning.

With this in mind, we lead with the problem of how the centralised module may provide such an information. To that end, we part from the notion of *informative mechanism* [4]. Those types of mechanisms are in charge of providing some kind of information to agents in order to regulate a multiagent system. Thus, an *informative mechanism* $\Gamma : \mathcal{S}' \times \mathcal{X}' \rightarrow \mathcal{I}$ is a function that given a partial description of an internal state of an agent (S') and, taking into account the partial view that the mechanism has of the current environmental state ($\mathcal{X}'$), provides certain information ($\mathcal{I}$). We adhere this definition to create mechanisms over the agreements for different situations, creating information valuable for participants in the organisation. Thus, all the information about reputation-based agreements reached within an organisation will be provided by means of informative mechanisms, formalised as follows:

Definition 4. *An informative mechanism providing information about reputation-based agreements is:*

$$\Gamma_{\Pi} : Sit \times \mathcal{X}' \rightarrow \mathcal{I}_{\Pi}$$

where:

- Sit is the situation an agent is requesting information about;
- $\mathcal{X}'$ is the environmental state;
- $\mathcal{I}_{\Pi}$ stands for the information provided by the mechanism by using the set of agreements Π reached over the situation Sit.

We have chosen a very general definition about information in order to cover all possible types of information the centralised module could offer taking into account the reputation-based agreements reached. The information provided may consist of a ranking sorting the best agents for a particular situation, such as $\langle \_, \mathcal{R}, \mathcal{A}, \_ \rangle$, created from the agreements reached for that situation, a value representing the reputation value for a situation, reached as consequence of the agreement for that situation, information about the properties of the agreement reached for a particular situation, for instance if it is full, complete, etc.

When we refer to situations as a key aspect when creating and using reputation-based agreements notice that instead of situation we could also follow the same processes to create reputation-based agreements about information related to organisational norms, such as: violation/fulfilment of norms. For instance, there could exist an informative mechanism providing a ranking with participants sorted by their degree of violation of certain norms. For the sake of simplicity we have described the terms of reputation-based agreement using situation, but exactly the same could be applied for organisational norms.

4 Case Study

In this section, we illustrate the proposed model by means of a simple case study. The scenario we use involves five different agents: *Anna*, *John*, *Jessica*, *Albert* and *Harry* participating within an organisation. In this organisation agents can *buy* and *sell* items, so the action space of agents is composed of actions such as, *buy-item-x*, *sell-item-x*, where x is whatever object they want to sell/buy. Besides, agents joined the organisation playing the following roles: *Anna* - *buyer*, *John* - *buyer*, *Jessica* - *buyer*, *Albert* - *seller* and *Harry* - *seller*. The situations in which an agent is involved in that organisation are regulated by organisational norms [5], some examples of such norms are:

- $\mathcal{ON}_1$: *"An agent playing the role seller must deliver the product sold 2 days after the payment, as maximum"*
- $\mathcal{ON}_2$: *"An agent playing the role buyer must pay 2 days after the purchase is performed, as maximum"*

After several interactions among them – performing actions of buying and selling different items – *Anna* decides to make public its opinion about *Albert* and *Harry* as sellers. Thus, she uses the reputation information messages to send to the centralised module its opinion, as follows:

$$\mathcal{R}^{info}_{Anna} = \langle\langle Albert, seller, \_, \_\rangle, 0.2\rangle$$
$$\mathcal{R}^{info}_{Anna} = \langle\langle Harry, seller, \_, \_\rangle, 0.9\rangle$$

This information shows that *Anna* has had bad experiences while she was buying things from *Albert* – 0.2 – maybe because *Albert* violated some organisational norm.[4]. Otherwise, she has had good experiences while she was buying things from *Harry* – 0.9 – maybe because *Harry* never violates organisational norms. Similarly, *John* and *Jessica* send their opinions about *Albert* and *Harry* as seller, by using the following reputation information messages:

$$\mathcal{R}^{info}_{John} = \langle\langle Albert, seller, \_, \_\rangle, 0.2\rangle$$
$$\mathcal{R}^{info}_{John} = \langle\langle Harry, seller, \_, \_\rangle, 0.8\rangle$$
$$\mathcal{R}^{info}_{Jessica} = \langle\langle Albert, seller, \_, \_\rangle, 0.2\rangle$$

It seems that both *John* and *Jessica* agree that *Albert* is bad selling items, however, *Harry* is good as a seller, from the point of view of *John*.

When the centralised module receives this information, it is able to create reputation-based agreements by using a function that aggregates the reputation information messages. Let us suppose that it aggregates the messages by calculating the average of reputation values sent by agents over exactly the same situation:

$$f_\pi(Sit) = \frac{\sum_{i=1}^{n} \mathcal{R}^{info}_{ag_i} = \langle Sit, RepVal_i\rangle}{n}$$

From the set of messages sent by the agents so far, the centralised module can create two reputation-based agreement regarding to two different situations:

$$\pi_1 = \langle\langle Albert, seller, \_, \_\rangle, \{Anna, John, Jessica\}, 0.2, t\rangle$$
$$\pi_2 = \langle\langle Harry, seller, \_, \_\rangle, \{Anna, John\}, 0.85, t\rangle$$

where the first component indicates the situation being evaluated, the second is the set of agents which have participated in the agreement, the third component is the reputation value agreed by the participants – it is calculated by using the function $f_\pi(Sit)$, i. e. it represents the average of all values sent about that situation –, and finally the fourth component is the time in which the agreement is reached.

Taking into account those agreements, the centralised module makes available three different informative mechanisms:

- $\Gamma^1_\Pi(\langle \mathcal{A}g, \mathcal{R}, \_, \_\rangle)$ given a situation where an agent and a role is specified, returns meta-information[5] about the reputation-based agreement reached about that situation;
- $\Gamma^2_\Pi(\langle \mathcal{A}g, \mathcal{R}, \_, \_\rangle)$ given a situation where an agent and a role is specified, returns the reputation-based agreement reached. In particular, it returns the reputation value in the agreement of that situation;

[4] We suppose that reputation values – denoted by $RepVal$ – are in the range [0..1]

[5] with meta-information we mean the α-consistency of the agreement, if it is full or complete

– $\Gamma^3_\Pi(\langle \_, \mathcal{R}, \_, \_\rangle)$ given a situation where a role is specified, returns a ranking of agents playing that role, sorted by the reputation value they have as consequence of the reputation-based agreement reached until the current time t.

Let us suppose that a new agent *Alice* join the organisation playing the role *buyer*. Since she does not know anybody within the organisation and she wants to buy something, she may use the informative mechanisms published to obtain information about other participants. For instance, *Alice* is looking for a *seller* to buy something, so a ranking of sellers will be a great solution to select the best one. Thus, she searches among all informative mechanisms if there exists one which provides that information[6]. She is very lucky finding Γ^3_Π, that returns a ranking when it is queried with a situation specifying a role. So, *Alice* performs the following query to Γ^3_Π:

$$\Gamma^3_\Pi(\langle \_, seller, \_, \_\rangle) \Rightarrow \{Harry, Albert\}$$

the informative mechanism returns a ranking of agents, sorted by the reputation values of all reputation-based agreements, where the situation specified in the query matches with the situation of agreements. In particular, the implementation of this mechanism searches among all agreements reached where the situation has the role *seller*. By using this information *Alice* knows that there exists an agreement within the organisation about *Harry* is better seller than *Albert*. But, how good are they?. To answer this question *Alice* queries the informative mechanism Γ^2_Π as follows:

$$\Gamma^2_\Pi(\langle Harry, seller, \_, \_\rangle) \Rightarrow 0.85$$
$$\Gamma^2_\Pi(\langle Albert, seller, \_, \_\rangle) \Rightarrow 0.2$$

Right now, *Alice* knows that *Harry* is better seller than *Albert* and there exists an agreement within the organisation that *Harry*'s reputation selling goods is 0.85 and another one that says that *Albert* as seller is 0.2 – in the range 0 and 1. However, *Alice* is still doubting about which seller could be the best, because she is wondering how consistent that agreement is. Thus, she wants to answer that question and she queries the informative mechanism that provides meta-information about the agreement reached regarding a situation. Therefore, she performs the following queries:

$$\Gamma^1_\Pi(\langle Harry, seller, \_, \_\rangle) \Rightarrow \pi^{0.05}$$
$$\Gamma^1_\Pi(\langle Albert, seller, \_, \_\rangle) \Rightarrow \pi^0$$

With this information *Alice* clears all her doubts, because now she knows that all opinions sent about *Albert* are coincident because of the reputation-based agreement reached is 0-consistent (π^0). Besides, she knows that the opinions sent by the agents that have interacted with *Harry* are almost the same since their variability is (0.05). With this in mind, *Alice*, finally, selects *Harry* as a seller.

[6] We suppose that informative mechanisms are published by the organisation to all participants

5 Discussion

There are several distributed reputation systems where the agents themselves are able not only to evaluate the behaviour of other agents and associate reputation values but they are able to aggregate different reputations related to different experiences. It is the case of the agents in Regret [11] that can aggregate reputation values created based on their individual experiences and also on reputations values reported by other agents.

As stated before, one of the main advantages of having a centralised reputation mechanism is feasibility for an individual to know a more consistent reputation about another agent based on numeral experiences. In the case of distributed mechanisms, the individual itself would need to participate in several interactions with the given agent and also to ask for other agents about their experience with it. In the case of the centralised mechanism, the agent can easily ask the *informative mechanisms* about the *reputation-based agreement* of the given agent in the desired situation.

In [8] the authors present an approach to create rankings able not only to provide the most trustful agents but also a probabilistic evidence of such reputation values. Those rankings are also computed by a centralised mechanism by aggregating the reputations reported by the agents. This approach and the one presented in our paper are complementary. This paper focuses on defining the ranking algorithms and ours focuses on describing the mechanism used to receive the reputation information and to provide the already evaluated agreements and rankings. Another work that is also complementary to ours is the one presented in [9]. They describe the algorithm *NodeRanking* that creates rankings of reputation ratings.

In order to motivate agents on reporting their experiences to the centralised mechanism several approaches can be used. Points can be provided when agents send reputation information to the mechanism and a given number of points can be required when the agent asks for reputation-based agreements or rankings. We assume that the agents know how important the information stored in the centralised mechanism is in order to them achieve their goals.

6 Conclusions

This work tries to put forward a novel approach of reputation-based agreement concept by supporting on a hybrid reputation model presented in [5]. This approach formalises a centralised module – complementary to the distributed mechanism presented in [5] – that defines reputation-based agreements as aggregations of participants' opinions sent to the module. We also define some properties that can be derived. Furthermore, we also propose to use the agreements reached by using the concept of informative mechanism [4], so providing agents with useful information based on those agreements. Some different examples are also given to clarify the importance of reaching reputation-based agreements and its utility for the participants in the organisation.

In future work we plan to experimentally test our approach by implementing a case of study. We are also interested in how to merge different agreements. Moreover, it would be interesting to study the aggregation of information sent in different periods.

References

1. Amazon. http://www.amazon.com, 2008.
2. H. Billhardt, R. Centeno, A. Fernández, R. Hermoso, R. Ortiz, S. Ossowski, J. Pérez, and M. Vasirani. Organisational structures in next-generation distributed systems: Towards a technology of agreement. *Multiagent and Grid Systems: An International Journal*, 2009.
3. C. Carrascosa and M. Rebollo. Modelling agreement spaces. In *IBERAMIA 2008 Workshop on Agreement Technologies (WAT 2008)*, pages 79–88. Marc Esteva, Adriana Giret, Alberto Fernández, Vicente Julián, 2008.
4. R. Centeno, H. Billhardt, R. Hermoso, and S. Ossowski. Organising mas: A formal model based on organisational mechanisms. In *24th Annual ACM Symposium on Applied Computing (SAC2009), Hawaii, USA, March 8-12*, pages 740–746, 2009.
5. R. Centeno, V. Torres da Silva, and R. Hermoso. A reputation model for organisational supply chain formation. In *Proc. of the 6th COIN@AAMAS'09 Budapest,Hungary*, pages 33–48, 2009.
6. J. Guedes, V. Silva, and C. Lucena. A reputation model based on testimonies. In *Agent Oriented Information Systems IV: Proc. of the 8th International Bi-Conference Workshop*, volume 4898 of *Lecture Notes in Artificial Inteligence*, pages 37–52. Springer-Verlag, 2008.
7. T. Huynh, N. Jennings, and N. Shadbolt. Fire: An integrated trust and reputation model for open multi-agent systems. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI)*, pages 18–22, 2004.
8. A. Ignjatovic, N. Foo, and C. Lee. An analytic approach to reputation ranking of participants in online transactions. In *WI-IAT '08: Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pages 587–590. IEEE Computer Society, 2008.
9. J. Pujol, R. Sangüesa, and J. Delgado. Extracting reputation in multi agent systems by means of social network topology. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 467–474, New York, NY, USA, 2002. ACM.
10. J. Sabater and C. Sierra. Reputation and social network analysis in multi-agent systems. In *Proceedings of First International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 475–482, 2002.
11. J. Sabater and C. Sierra. Review on computational trust and reputation models. *Artificial Intelligence Review*, 24(1):33–60, 2005.
12. V. Silva, R. Hermoso, and R. Centeno. A hybrid reputation model based on the use of organization. In J. Hubner, E. Matson, O. Boissier, and V. Dignum, editors, *Coordination, Organizations, Institutions, and Norms in Agent Systems IV*, volume 5428 of *LNAI*. Springer-Verlag, 2009.
13. B. Yu and M. Singh. Distributed reputation management for electronic commerce. *Computational Intelligence*, 18(4):535–549, 2002.

Towards an abstract architecture for service discovery with semantic alignment

Analay Baltá, Alberto Fernández

CETINIA, University Rey Juan Carlos, Móstoles, Spain
analay@ia.urjc.es, alberto.fernandez@urjc.es

Abstract. In large-scale open environments mechanisms for locating appropriate services have to deal with the additional problem of semantic mismatches among the components. Semantic alignment mechanisms need to be purposefully integrated into a service discovery framework in order to fully exploit its potential. The objective of this paper is to present an ongoing work towards the analysis and design of basic mechanisms able to locate adequate services in open heterogeneous environments. An abstract architecture that addresses the semantic mismatches at service description model level as well as domain ontology is presented. Several open issues are pointed out.

Keywords: service oriented architecture, semantic web services, service discovery, matchmaking,

1 Introduction

In multiagent systems, agents communicate with the aim of achieving their objectives. Agents are autonomous entities capable of planning the tasks they have to carry out to maximize their utilities. An individual agent may require a service to be performed by another entity. In order to be able to achieve a fruitfully interaction the two agents must understand the semantic of the messages they exchange. This is typically done by sharing the same ontology, although this is not easy to achieve in open systems. Another option is to use ontology bridges, which make use of ontology alignment techniques [1, 2] to transform information from one ontology to another.

There are several stages since an agent identifies a given need until the service that provides it is eventually executed. First, the agent identifies some functionality that it is not able to perform or that might be executed more efficiently by an external entity. Then, candidate service providers must be located. Once a set of potential providers are known, the agent must choose one among them. This decision can be made based on several factors such as quality of service, price, reputation, etc. After the selection is made the two agents might engage in a negotiation about the conditions under which the service is going to be performed. After an agreement has been reached the service can be called. *Agreement Technologies* [3] like semantic alignment, negotiation, argumentation, virtual organizations, decision making, learning, trust, and so on, will be used to develop such large-scale open systems.

We concentrate on the phase of provider location. Distributed service directories and efficient decentralised matching techniques with powerful description languages are essential for dynamic and scalable service discovery. Furthermore, in such environments the mechanisms for locating appropriate services have to deal with the additional problem of semantic mismatches among the components.

In this paper we present an ongoing work towards the development of a service discovery framework where semantic alignment mechanisms are purposefully integrated into.

The rest of the paper is organized as follows. In the next section we begin by describing the Service Oriented Architecture so as to establish the context in which this work is situated. Section 3 presents an abstract architecture for semantic service discovery, which pays especial attention to semantic alignments of service descriptions. In section 4 we discuss several open issues for further research. Finally, we present some conclusions and future work.

2 Service Oriented Architectures

Fig. 1 shows a general Service Oriented Architecture (SOA) [4]. It involves three or more parties: a *requester,* one or more *providers* and a *directory*, which supports services during the transaction and possibly mediates between the requester and the provider. Roughly speaking, the requester corresponds to the client, and the provider corresponds to the server in the typical client-server architecture.

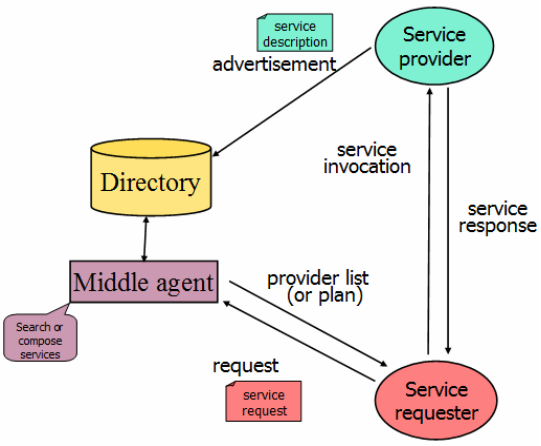


Fig. 1 General Service Oriented Architecture.

Web Services are the reference enabling technology for SOA. Web Services can be seen as a collection of technologies, protocols and standards that build programming solutions for specific application integration problems. As the number of available web services is steadily increasing, companies realize the need for automatically discovering web services and having an automated composition.

SOA combines the service discovery, selection, and engagement, thus adding a new level of functionality on top of the current Web. The addition of semantic information to describe Web Services, in order to enable the automatic location, combination and use of distributed components, is nowadays one of the most relevant research topics due to its potential to achieve dynamic, scalable and cost-effective Enterprise Application Integration and eCommerce.

The process of discovering and interacting with a Semantic Web Service includes the following phases [5]:

(i) Candidate Service Discovery is the distributed search for available services that can accomplish the client's internal goal or objective. It is a process of identifying candidate services by clients. It involves three types of stakeholders: service *providers* that publish service advertisements, service *requesters* that require a service and *matchmakers* that accept descriptions of available service from providers and match them against requirements from requesters.

(ii) Service Engagement includes the process of interpreting candidate service enactment constraints, described by each candidate service published, and then requesting or possibly negotiating with prospective services to reach an agreement. Engagement concludes with both service and client knowing and agreeing to the terms of service provision in an explicit or implicit service contract.

(ii) Service Enactment consists of alternative protocols to initiate service activity, monitor service processes, and confirm service completion. If the service terminates abnormally after a contract has been formed, there may be a final set of protocol interactions to address compensation issues.

Proper methods to enable the automatic location and selection of suitable services in order to solve a given task or user request are an essential ingredient. To this end, several description frameworks to annotate provided services on the one hand and express service requests on the other have been proposed. They range from logic-based complex and expressive semantic service descriptions (e.g. OWLS [6], WSMO [7]) to syntactical ones (WSDL [8], keywords, tag clouds), with some approaches in between (SAWSDL [9]). Semantic service descriptions are supported on ontologies.

In this context, several description frameworks to semantically match services on the one hand and service requests on the other have been presented in the literature. Many of the current proposals for defining the degree of match between service advertisements and requests are based on subsumption checking of concepts present in inputs and outputs of service descriptions.

3 Abstract Service Discovery Architecture

Fig. 2 illustrates the proposed architecture that defines the service discovery functionality. The architecture comprises the building components to match a service request against a service advertisement. In particular, this proposal pays special attention to the problem of semantic mismatches between descriptions. Semantic mismatches are considered at two different levels:

- *Service description models.* Services (advertisements and requests) might be described using different languages or models (e.g. OWL-S, WSMO, SAWSDL, ...).
- *Domain ontology concepts.* Since semantic service descriptions rely on the use of domain ontologies, the second type of mismatch is due to the use of different domain ontologies to specify the concepts used in the descriptions.

Note that both options can be combined. For instance, two services might share the same service model (e.g. OWL-S) but use different domain ontologies, or they might use the same domain ontology but different service models.

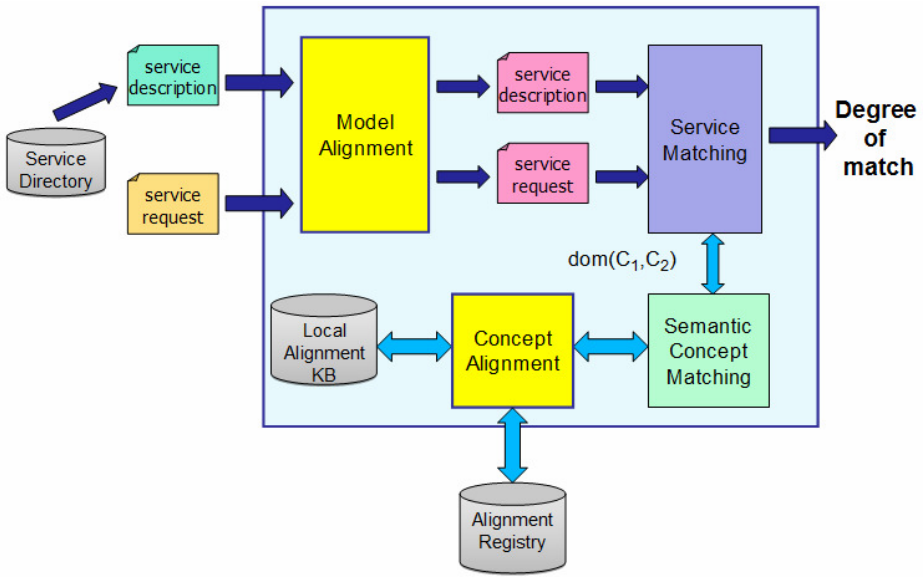


Fig. 2 Service Discovery Architecture

The first step in the matching of two services is the *alignment of the service description models*. Note that service description approaches not only differ in the language in which they are written. They are classified at different levels of expressiveness, ranging from complex, formal, logic-based semantic descriptions to lightweight syntactical ones. Service model alignment consists of mapping both service descriptions (request and advertisement) into a common service model.

Once the adequate model alignment has been applied, the unified service descriptions are prepared to be matched. The *Service Matching* module takes those two descriptions and returns the degree of match between them. We envision here the existence of different matchmakers to deal with different common models. Matchmaking algorithms usually include a *Semantic Concept Matching* process to analyze the (similarity) relation between concepts used in advertisements and requests. As it was pointed out above, those concepts might belong to different ontologies so a *Concept Alignment* is carried out to solve this problem. In this case, we keep a local knowledge base of alignments and assume the existence of a external registry of alignments. The *local knowledge base* acts as a cache of alignments used in previous matchings. The *external alignment registry* is consulted to avoid carrying out a process of ontology alignment if there is an alignment published by third parties.

In the next sections we go into details of the building blocks of the proposed architecture.

3.1 Service Model Alignment

As commented above, service model alignment consists of defining a common model for two different ones, and mapping descriptions described in those source models into the common model. This transformation may produce a loss of expressiveness in at least one of the original descriptions, especially if they use models with different expressiveness power. Here, we do not aim at the design of a unified model for any description, which would probably lead us to the definition of a very simple (lightweight) model to account for all the different source models. However, we envision the definition of mappings between pairs of models, thus keeping that particular common model as close to the original ones as possible. Besides, this approach is more modular and flexible to consider new models. Model-to-Model alignments consist of three steps:

1. *Conceptual analysis* of characteristics finding mappings between models. Note that this task can be done focusing on service matchmaking, i.e. only the aspects considered by the matching techniques have to be mapped.
2. Definition of a *common model language* (CML), which might be one of the originals. The use of standard languages is encouraged here.
3. Implementation of a tool for *automatic transformation* of service descriptions from the original models to the CML.

In the next subsection we show an example of the alignment between the two most important Semantic Web Service description languages, OWL-S and WSMO.

3.1.1 WSMO/OWL-S alignment

In order to establish the relationships between the terminologies used in each ontology and propose a mapping, we set out from existing conceptual comparisons between WSMO and OWL-S [10, 111]. As shown in Table 1, Web Service descriptions in WSMO are defined by their *preconditions, posconditions, assumptions* and *effects*, and their equivalent in OWL-S are *inputs, outputs, preconditions* and *results*, respectively.

Table 1. Mapping between OWL-S and WSMO.

Element	OWL-S	WSMO
input	hasInput	precondition
output	hasOutput	postcondition
precondition	hasPrecondition	assumption
effect	hasResult	effect
others		

After the conceptual mapping between models, the next step is the definition of a common description language. We opt for using RDF as the common model language. RDF is a W3C recommendation language to represent resources on the Web. There are a lot of RDF contents and tools to process them. Although RDF is less expressive than OWL-S and WSMO, it is enough (in practice) for representing the information needed for service search. It also allows the use of SPARQL [12] to query the descriptions. The resulting RDF Schema graph describing semantic Web services for this mapping is shown in Fig. 3 and the RDF code is shown in **Fig. 4**.

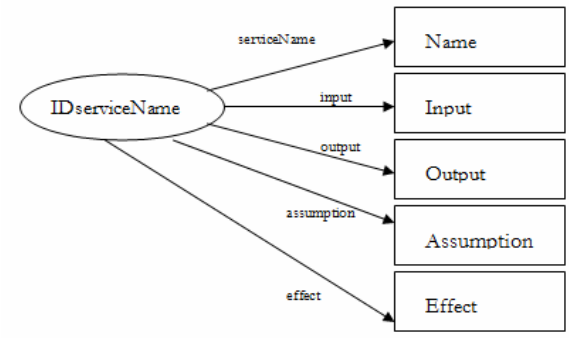


Fig. 3. RDF graph for service discovery

```
<?xml version="1.0"?>
<rdf:RDF>
  < rdf: Description rdf:ID="IDserviceName">
  </rdf:Description>
  <rdf:Description rdf:ID="serviceName">
  <rdfs:domain rdf:resource="#IDserviceName"/>
  <rdfs:range rdf:resource="#Name"/>
  </rdf:Description>
  <rdf:Description rdf:ID="input">
  <rdfs:domain rdf:resource="#IDserviceName"/>
  <rdfs:range rdf:resource="#Input"/>
  </rdf:Description>
  <rdf:Description rdf:ID="output">
  <rdfs:domain rdf:resource="#IDserviceName"/>
  <rdfs:range rdf:resource="#Output"/>
  </rdf:Description>
  <rdf:Description rdf:ID="assumptions">
  <rdfs:domain rdf:resource="#IDserviceName"/>
  <rdfs:range rdf:resource="#Assumptions"/>
  </rdf:Description>
  <rdf:Description rdf:ID="effect">
  <rdfs:domain rdf:resource="#IDserviceName"/>
  <rdfs:range rdf:resource="#Effect"/>
  </rdf:Description>
</rdf:RDF>
```

Fig. 4. RDF/XML representation of the CML for the OWL-S/WSMO alignment

3.2 Service Matching

Many of the current approaches to Semantic Web Services matching, particularly those based on OWL-S, started from the work of Paolucci et al. [13]. This approach proposes a matching algorithm that takes into account inputs and outputs of advertised (A) and requested (R) services. A match between two output concepts (O_A, O_R) is the (subsumption) relation between the two concepts in the ontology. They differentiate among four degrees of match: *exact* ($O_A = O_R$), *plug-in* (O_R subsumes O_A), *subsumes* (O_A subsumes O_R) and *fail* (otherwise). An output matches if and only if for each output of the request there is a matching output in the service description. If there are several outputs with different degree of match, the minimum degree is used. The same algorithm is used to compute the matching between inputs, but with the order of request and advertisement reversed. Finally, the set of service advertisements is sorted by comparing output matches first, if equally scored, considering the input matches. Several authors extend or propose variations to that proposal (e.g.[14,15]).

Several similarity (or distance) measures for concept matching have been proposed in the literature, although their application to the concrete domain of service matching is very limited. One of the most well known distance measures between concepts is the length of the shortest path between them in the taxonomy, proposed by Rada et al [16]. Other proposals further refine that approach ([17, 188]). Other authors do not base concept similarity on the distance between the concepts ([19, 20, 21]).

We propose a combination of service matching and concept similarity. In [22] we describe how both approaches can be combined into a unified service selection framework which returns a numeric value that can be used for ranking services. The

ranking function compares the level of match first (e.g. exact, plugin, ...), and then the level is refined with the (numerical) similarity value. Since service descriptions consist of several components (inputs, outputs, ...), the similarity between services must be defined based on its individual elements (e.g. each of its inputs) and aggregation operators.

3.2.1 SPARQL as service query language

If we envision the representation of services in RDF (like the proposed common model for OWL-S and WSMO), then SPARQL [12] might be used as a query language for services. Note that the SPARQL query might be created either from the scratch or by transforming a service request. In this section we analyse how those requests look like, and the potential and limitations of SPARQL as service request language. We use the RDF service descriptions proposed in section 3.1.1.

This is an example of a first attempt of defining a flight service request in SPARQL:

```
SELECT ?Name
WHERE {?x serviceName ?Name .
       ?x input #DepartureAirport .
       ?x input #ArrivalAirport .
       ?x input #OutboundDate .
       ?x input #InboundDate .
       ?x output #FlightsFound .
       ?x output #PreferredFlightItinerary .
       ?x effect #HaveSeatResult}
```

This query will return all service names that have at least the four inputs, the two outputs and the effect specified in the query (conditions in the *WHERE* section are interpreted as conjunctions).

Remember that a service advertisement (*A*) match a request (*R*) if and only if all the inputs of *A* are provided by *R* and all the outputs of *R* are provided by *A*.

The first problem we identify is that services that only need a subset of the specified inputs would not be returned. However, services providing more outputs than needed by the requester cause no problem. This problem can be solved by decomposing the query in two: (i) querying the advertisements using the outputs and (ii) using the results to query the original request about the inputs.

The next problem identified is that querying RDF graphs only returns *exact* matches. The use of an inference engine to classify the ontology (compute *subclass* relations) or to interpret the query would provide subsumption reasoning, thus supporting the matchmaking using levels of match (*exact, plugin, subsumes, ...*).

However, if we need more refined complex matching functions, e.g. taking into account the distance between concepts in a taxonomy, there is no straight way to do it. Instead, new SPARQL inference engines adapted for service matchmaking should be developed.

Finally, we have to devise how to include semantic alignments into this approach. A solution could go in the line of query transformation by including fragments to consulting RDF alignment specifications (see section 3.3).

3.3 Concept Alignment

In the previous section we saw that current service matchmaking algorithms are based on checking the relations between the concepts that appear in the different fields of semantic service descriptions. If the concepts being compared are defined in different ontologies then semantic alignments must be considered instead of obtaining a *fail* match.

An alignment (or mapping) between two ontologies *O* and *O'* can be described as a quadruple [23]:

$$<e, e', n, R>$$

where:

- *e* and *e'* are the entities between which a relation is asserted by the mapping (e.g., formulas, terms, classes, individuals)
- *n* is a degree of trust (confidence) in that mapping
- *R* is the relation associated to a mapping, where *R* identifies the relation holding between *e* and *e'*.

In this work we are not concerned about ontology alignment techniques, but on the use of alignments. Thus, we are interested in representing and querying mappings between ontologies. We propose using RDF as the language for expressing alignments, so that they can be published on the web and queried using SPARQL. In particular, we use the format of the Ontology Alignment Evaluation Innitiative[1].

4 Open Issues

In addition to the aforementioned aspects, we point out here some additional open issues that will need to be dealt with.

The abstract architecture proposed in this paper describes the process of obtaining the degree of match between a service advertisement and a service request. Typically, this process is carried out by a middle agent or matchmaker (either separated or integrated in the directory of services). Roughly, that process takes as inputs two service descriptions and returns a matching degree (a level of match or a numerical value).

However, there might be situations in which this task should be done in a different way. Firstly, it is arguable that the matching process can be effectively done in a one step process. As Lara point out [24], the semantic description capability of services and of customer goals can be exploited by the service discovery based on a two-phased service discovery model. The first phase identifies services that can provide results required by the customer or specifies semantic matchmaking on the goal template level. In the second phase, the input values required by relevant services are considered and only services for which the customer can provide appropriate input, and for which this input can lead to the expected results are selected. The input values will partially determine the results of the service. Then there is a combination of query of selected service and answer of customer. In the best case the customer can

[1] http://oaei.ontologymatching.org/

only guess, when defining his goal, what input values will be required by services satisfying such goal. In addition, the customer might have a considerable volume of information from which input values to services can be obtained.

Another usual convention is that service matchmaking is completely carried out in the middle agent (or directory) side. This is possible under the assumption that the middle agent has all the necessary information, i.e. it has complete service descriptions. However, under some circumstances, the provider or the requester might not be interested in revealing some private data, i.e. a credit card number. In those contexts, some mechanisms are needed to deal with such sensitive information. One possible option is to carry out part of the matching on the requester and/or the provider side. In these cases, efficiency issues have to be considered.

In the development of methods, techniques and tools for open large scale environments, scalability issues are fundamental. In the context of service discovery and the architecture presented in Fig. 2, there are two main points that must be considered. They are the decentralised service directories and the discovery of ontology alignments. In both cases, the recent trends in querying distributed data on the semantic web by means of SPARQL end points will be considered in future research.

As described previously, services can be described in different languages at several levels of expressiveness. Depending on the context in which the searched service is going to be used, the set of candidate services can be larger or smaller. For instance, if the service is expected to be used for automatic invocation or for composition, only semantic service descriptions (OWL-S, WSMO, SAWSDL, ...) would be considered. However, if the service is expected to be used by a human user then also more lightweight descriptions (e.g. keyword-based) would be considered.

5 Conclusion

In this paper we have dealt with the problem of service discovery in open systems. We proposed an abstract architecture that has semantic alignment as a first citizen component. We provided preliminary ideas and developments towards the construction of a service discovery framework in which semantic alignment mechanisms are purposefully integrated into.

In particular, we discussed in detail the alignment of OWL-S and WSMO, their differences and the transformation of both into a RDF common model. We also proposed the combination of service matching and concept similarity into an integrated service matching framework. We analysed the use of SPARQL as a service query language and identified the pros and cons, and RDF as ontology alignment representation format.

The definition of other service description model alignments as well as the implementation of the proposed framework are part of our ongoing work. In the future we also plan to investigate the issues pointed out in section 4 (two-phase service discovery, distributed service directories and context).

Acknowledgment: This work has been partially supported by the Spanish Ministry of Science and Innovation through grants TIN2006-14630-C03-02, and CSD2007-0022 (CONSOLIDER, INGENIO 2010)

6 References

1. Jérôme Euzenat and Pavel Shvaiko. Ontology matching. Springer, Heidelberg (DE), 2007
2. Marc Ehrig: Ontology Alignment: Bridging the Semantic Gap. Springer. 2007.
3. Agreement Technologies. www.agreement-technologies.org
4. Papazoglou, M., Van den Heuvel, W. Service oriented architectures: approaches, technologies and research issues. Springer-Verlag 2007.
5. Burstein, M., Bussler, C., Zaremba, M., Finin, T., Huhns, M. N., Paolucci, M., Sheth, A. P., and Williams, S. A Semantic Web Services Architecture. *IEEE Internet Computing* 9, 5, 72-81. 2005.
6. Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDemott, D., McIlraith, D., Narayanan, D., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., and Sycara, K.. OWL-S: Semantic Markup for Web Services. W3C Member Submission, 2004. Available from http://www.w3.org/Submission/OWL-S/.
7. Bruijn, J., Bussler, C., Domingue, J., Fensel, D., Hepp, M., Keller, U., Kifer, M., K¨onig-Ries, B., Kopecky, J., Lara, R., Lausen, H., Oren, E., Polleres, A., Roman, D., Scicluna, J., and Stollberg, M. Web Service Modeling Ontology (WSMO). W3C Member Submission, 2005. http://www.w3.org/Submission/WSMO/.
8. E. Christensen, F. Curbera, G. Meredith, and S.Weerawarana. Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl, March 2001
9. Joel Farrell and Holger Lausen. Semantic Annotations for WSDL and XML Schema (SAWSDL). W3C Recommendation 28 August 2007. http://www.w3.org/TR/sawsdl/
10. Polleres, A., Lara, R. A Conceptual Comparison between WSMO and OWL-S, WSMO Working Group working draft, 2005. http://www.wsmo.org/2004/d4/d4.1/v0.1/.
11. Lara, R.., Polleres, A. D4.2v0.1 Formal Mapping and Tool to OWL-S, WSMO working draft 17 december 2004. http://www.wsmo.org/2004/d4/d4.2/v0.1/
12. W3C World Wide Web Consortium. SPARQL Query Language for RDF. W3C Recommendation 15 January 2008. http://www.w3.org/TR/rdf-sparql-query/.
13. Paolucci, M., Kawamura, T., Payne, T., and Sycara, K. Semantic Matching of Web Service Capabilities. In ISWC, pages 333–347. Springer Verlag, 2002.
14. Klusch, M., Fries, B., and Sycara, K. Automated semantic web service discovery with owls-mx. In AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, pages 915–922, New York, NY, USA, 2006. ACM Press.
15. Li, L and Horrocks, I. A software framework for matchmaking based on semantic web technology. Int. J. of Electronic Commerce, 8(4):39–60, 2004.
16. Rada, R., Mili, H., Bicknell, E., and Blettner, M. Development and application of a metric on semantic nets. IEEE Transactions on Systems, Man and Cybernetics, 19(1):17–30, 1989.
17. Leacock, C. and Chodorow, M. Combining local context andWord- Net similarity for word sense identification. In Christiane Fellbaum, editor, WordNet: An Electronic Lexical Database, pages 265–283. MIT Press, 1998.
18. Wu, Z. and Palmer, M. Verbs semantics and lexical selection. In Proceedings of the 32nd annual meeting on Association for Computational Linguistics, pages 133–138, Morristown, NJ, USA, 1994. Association for Computational Linguistics.

19. Resnik, P. Using information content to evaluate semantic similarity in a taxonomy. In IJCAI, pages 448–453, 1995.
20. Borgida, A., Walsh, T., and Hirsh, H. Towards measuring similarity in description logics. In Description Logics, 2005.
21. Noia, T., Di Sciascio, E., Donini, F., and Mongiello, M. Semantic matchmaking in a p-2-p electronic marketplace. In SAC, pages 582–586. ACM, 2003.
22. Fernandez, A., Polleres, A., and Ossowski, S. Towards Fine-grained Service Matchmaking by Using Concept Similarity, ISWC-2007 Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web (SMR2). 2007
23. Paolo Bouquet, Jérôme Euzenat, Enrico Franconi, Luciano Serafini, Giorgos Stamou, and Sergio Tessaris. Specification of a common framework for characterizing alignment. deliverable D2.2.1, Knowledge web NoE, 2004.
24. Lara, R. Two-phased web service discovery. In AI-Driven Technologies for Services-Oriented Computing workshop at the Twenty First National Conference on Artificial Intelligence (AAAI-06), Boston, USA, 07/2006 2006.

A simulator for a two layer MAS adaptation in P2P networks

Jordi Campos[1], Marc Esteva[2], Maite López-Sánchez[1] and Javier Morales[2]

[1] MAiA Deptartment, Universitat de Barcelona, email: {jcampos,maite}@maia.ub.es
[2] Artificial Intelligence Research Institute (IIIA), CSIC,
email:{marc,jmorales}@iiia.csic.es

Abstract. Adapting organisational structures to maintain an organisation effectiveness under varying circumstances is becoming a hot topic within the agent community. In this paper we present a simulator that we have developed for testing adaptation mechanisms in Peer to Peer scenarios. We regard this service as part of the new generation of services that should be incorporated into multiagent systems infrastructures to assist coordination both at participant and organisation levels. In order to provide such organisational adaptation we rely on an added distributed meta-level. Meta-level agents perceive partial information about system properties that they use to adapt organizational structures when necessary. The simulator implements different sharing methods, allows to define different network topologies, and includes some facilities to process and analyse simulation results in order to compare them.

1 Introduction

Organisational structures have proven to be useful to regulate MultiAgent Systems (MAS) [1, 2]. However, certain environmental or population changes may imply a decrease in goal fulfilment. Thus, adapting organisations is now becoming an active research area [3–6], since it can help to keep the expected system outcomes under changing circumstances.

In particular, we propose to add a *meta-level* in charge of adapting system's organisation instead of expecting agents to increase their behaviour complexity. This is specially relevant when dealing with open MAS, since there is no insight of participant's implementation, and hence, we can not guarantee that agents are endowed with organisational adaptation capabilities. We regard this adaptation –together with other possible *meta-level* functionalities– as an assistance to agents that can be provided by MAS infrastructure. Thus, we call our implementation approach Two Level Assisted MAS Architecture (2-LAMA)[7]. Furthermore, in order to avoid centralisation limitations such as fault-tolerance or global information unavailability, we propose this architecture has a distributed *meta-level* —i.e., it is composed of several agents. In this context, in this paper we present a simulator for testing organisational adaptation mechanisms in P2P scenarios. Hence, the main goal of the paper is to describe the simulator functionality.

Our approach is able to deal both with highly dynamic environments and domains without direct mapping between goals and tasks —i.e. domains where it is not possible to derive a set of tasks that achieve certain goals. Nevertheless, it requires domains where organisations can be dynamically changed. Peer-to-Peer sharing networks (P2P) present all previous features, and thus, we use them as a case study. In such networks, computers contact among them to share some data and their relationships change over time depending on network status and participants. In this context, a P2P system is modelled as an organisation having a social structure among peers and a set of protocols and norms that regulate the sharing process. On top of the P2P system –that we call domain-level– we add a distributed meta-level that perceives status information and uses it to adapt peers' social structure and norm values. Meta-level adaptation is based on system performance, which is measured by the time peers spent to share data and the required network consumption.

This paper is structured as follows. Next section provides a more detailed description of the 2-LAMA model, so that Section 3 can apply it to the P2P scenario. This scenario has been implemented in the simulator presented in Section 4. Finally, some conclusions and future work are described in last section.

2 2-LAMA Model

Organisational structures regulate MAS by providing an agent coordination framework and some domain independent services that alleviate agent development. We regard these services as *Coordination Support* services [8] that include basic coordination elements such as elemental connectivity or agent communication languages. Usually, these services are devoted to enact agent coordination at different levels. In addition to them, we propose new services that provide an added value by assisting coordination further than just enabling it. In this manner, we group services in two different layers: an *Organisational Layer* that provides coordination enabling services, and an *Assistance Layer* on top of it, which provides coordination assistance services. This last layer includes a proactive service that adapts organisations depending on system's evolution.

The *Organisational Layer* provides basic services supporting the enactment of the organisation. We denote an organisation as $Org = \langle SocStr, SocConv, Goals \rangle$ where:

- *SocStr* corresponds to the social structure, which consists of a set of roles and the relationships among them.
- *SocConv* are social conventions that agents should conform and expect others to conform [9]. They are expressed as norms and/or interaction protocols, which define legitimate sequences of actions performed by agents playing certain roles.
- *Goals* describe the organisation design purpose —as opposed to agents' individual goals. They are expressed as a function on certain observable properties so that the system can evaluate its own performance.

Regarding our *Assistance Layer*, it provides two main types of services [8]: *Agent Assistance* and *Organisational Assistance*. The former assists individual agents to follow current social conventions. It includes four different services: *Information* that is useful for participating in the MAS; *Justification* of specific actions' consequences; *Advice* of alternative plans conforming social conventions; and *Estimation* of action consequences due to current conventions. The latter, the *Organisational Assistance*, consists on adapting the existing organisation in order to improve system's performance under varying circumstances. To provide such an adaptation, we propose goal fulfilment as its driving force within the context of a rational world assumption. Hence, the *Assistance Layer* requires some way (i) to observe system evolution, (ii) to compare it with the organisational goals and (iii) to adapt the organisation trying to improve goal fulfilment.

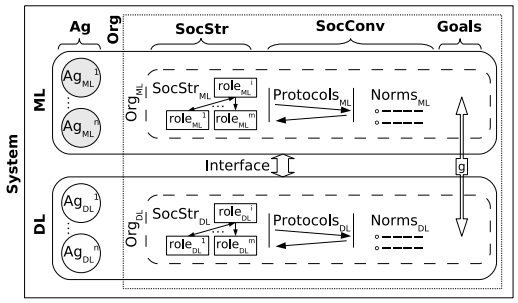


Fig. 1. Two Level Assisted MAS Architecture(2-LAMA): Domain Level (DL), Meta Level (ML) and Interface.

In order to provide *Assistance Layer*'s services, we have proposed a *Two Level Assisted MAS Architecture* (2-LAMA, [7]). It consists on a distributed *meta-level* (ML) that provides assistance to a *domain-level* (DL) in charge of domain-specific tasks. Figure 1 shows them and their communication trough an interface (Int). Thus, the whole system can be expressed as $2LAMA = \langle ML, DL, Int \rangle$[3]. Each level has a set of agents with its own organisation: $DL = \langle Ag_{DL}, Org_{DL} \rangle$ and $ML = \langle Ag_{ML}, Org_{ML} \rangle$. Using the interface, ML agents perceive partial information[4] about environmental observable properties ($EnvP$, e.g. date or temperature) and agents' observable properties (AgP, e.g. colour or position). In particular, a ML agent has partial information about the subset of DL agents it assists. We assume DL agents are grouped into clusters according to a domain-specific criterion —e.g. interaction costs. Therefore, a ML agent –we call it *assistant*– assists a cluster of DL agents, observes partial information about them, and shares it with other ML agents in order to provide better assistance services.

[3] It is possible to nest subsequent *meta-levels* updating previous level's organisation.
[4] In many scenarios global information is not available.

3 P2P model

Our case study is a simplified version of real Peer-to-Peer sharing networks (P2P), where a set of computers connected to the Internet (peers) share some data. This setting represents a highly dynamic and complex scenario, and thus, it is suitable for the development of a simulator implementing our approach.

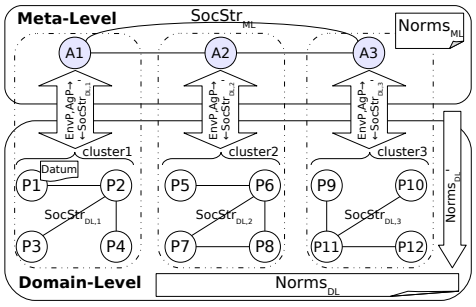


Fig. 2. 2-LAMA in the P2P scenario. Agents: peers P1..P12 at DL and assistants A1..A3 at ML.

Figure 2 shows our P2P model, where the DL is composed by agents playing the peer role. DL *social structure* determines agents' relationships, which corresponds to the neighours peers contact to share the data. The organisational goal (*Goals*) is that all peers receive the data with the minimal time and network consumptions. Hence, the time needed to share the data and the network consumption are the two metrics to evaluate simulation results. The *social conventions* at DL include the sharing protocol specified below and two norms. First *norm* limits agents' network usage in percentage of its nominal *bandwidth*.This *norm* can be expressed as: $normBW_{DL}$ ="a peer cannot use more than $\texttt{max}_{\texttt{BW}}$ *bandwidth* percentage to share data". This way, it prevents peers from massively using their *bandwidth* to send/receive data to/from all other peers. Second *norm* limits the number of peers to whom a peer can simultaneously send the data. Hence, we define $normFriends_{DL}$ ="a peer cannot simultaneously send the data to more than $\texttt{max}_{\texttt{Friends}}$ peers".

As we have already mentioned, ML provides assistance to DL. Each ML agent plays the *assistant* role for a cluster of DL agents (peers). It does it so by collecting information and adapting their local organisation. Its decisions are based on local information about its cluster, aggregated information about other clusters and the norms at ML. Some examples of local information are latencies ($EnvP$) or which peers have the data (AgP). Information about other clusters come from other *assistants* in the ML *social structure*. As for ML norms, we consider one limiting the number of peers –in the *cluster*– to inform about a new peer –in another cluster– having the data. Thus, we define $normHas_{ML}$ ="Upon reception of a completed peer (p $\notin$ *cluster*) message, inform no more than $\texttt{max}_{\texttt{Has}}$

Phase	Level	Protocol Messages
initial	ML	join<hasDatum>
latency	ML	get_latency<peers>, latency<peer><measure>
	DL	lat_req, lat_rpl
social structure	ML	contact<peers>
handshake	DL	bitfield<hasDatum>
data sharing	DL	request, data, cancel
	ML	completed, completed_peer<peer>, has_datum<peer>, all_completed
inactive	DL	have
waiting	DL	choke, unchoke
norms	ML	suggested_bw<value>, suggested_friends<value>, norm_updated<norm_id><new_definition>

Table 1. Protocol messages grouped into subsequent phases.

peers $\in$ *cluster* ". Finally, we assume *assistants* are located at Internet Service Providers (ISP), and thus, related communications are fast.

3.1 Protocol

Our proposed protocol is a simplified version of the widely used BitTorrent [10] protocol[5]. Table 1 presents the messages that are exchanged during protocol phases. Notice that the table includes the messages among DL agents, but also messages involving assistants at ML. Initially, peers join their cluster by informing its assistant. Afterwards, in order to compute the *social structure*, assistants need local information and therefore, they initiate latency phases requesting peers to measure their latency with all other peers in their clusters. Assistants use this information to propose a social structure among peers in their clusters. The social structure defines the overlay network within a cluster —i.e. which peers each peer has to contact in order to obtain the data.

Thereafter, peers perform a handshake phase where they introduce themselves to their contacts, and specify whether they have the datum. If this is the case, a data sharing phase starts —including data request and data transmission. Otherwise, as soon as one peer receives the datum, it will inform its handshaked peers so that the sharing phase is triggered this time. Nevertheless, upon request, a source peer cannot start a transmission if it is already serving the maximum number of allowed peers (defined by the $\texttt{max}_{\texttt{Friends}}$ value). For those cases, transmission can only be initiated when a previous transmission ends.

Upon data reception, a peer also informs its assistant. Then, this assistant shares this information with other assistants, who, in turn, inform some ($\texttt{max}_{\texttt{Has}}$) peers, so new data sharing phases can be started. An assistant also informs other assistants when all peers in its cluster are completed preventing further unnecessary communications. Finally, norm deliberations and notifications also belong to the protocol.

[5] Specially, we assume the information is composed of a single piece of data.

4 Simulator

As a platform to run our experiments, we have implemented a P2P sharing network simulator in Repast Simphony. Its architecture allows to both model agents (*agent-level*) and the transport of messages among them (*network-level*). The model that simulates the message transport is a packet switching network. Simulation at network level allows us to compare different P2P approaches taking into account the environmental changes that occur at this level (notice that this feature is not present in Repast Simphony simulator framework). In our current implementation, network status just depends on MAS activity, but we could introduce additional traffic that disrupts it. Our simulator includes our 2-LAMA approach and the standard BitTorrent protocol, and provides some facilities to collect information at agent/network level and to analyse it.

4.1 Architecture

Our simulator has an internal architecture that clearly isolates different functionalities. On the one hand, we have a module called `p2p` that represents the conceptual model defined by the 2-LAMA and is targeted to drive the simulation at *agent-level*. It provides tools to create state-based agents, to define a problem (number of peers, who has initially the datum, etc.) or services such as an agents' directory. The upper part of figure 3 shows the P2P implementation of the 2-LAMA architecture described in previous section.

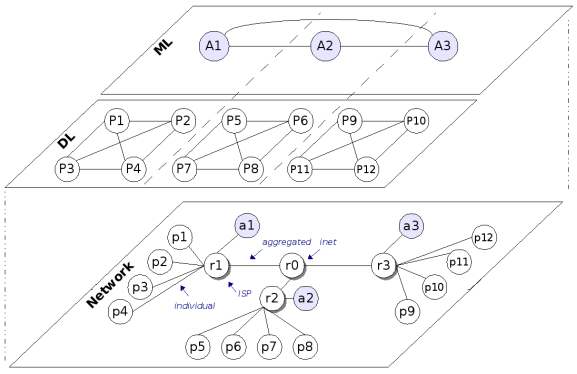


Fig. 3. 2-LAMA model and the underlying network. Agents are modelled on the top but their exchanged messages traverse the network on the bottom.

On the other hand, we have a module that drives the simulation at network-level. This module is called `netsim` and provides facilities to transport messages among agents, to define different network topologies, and to collect statistical

information about network status. In order to use this module, an agent from the `p2p` module is attached to a `netsim`'s network adapter, which is in charge of actually sending messages. These messages are split into packets that travel along links and follow their path by switching at routers. The destination agent is informed when each packet reaches its network adapter. Eventually, when all packets of a message arrive, the network adapter also delivers the whole message to the destination agent. Hence, agents can pay attention to packets or just wait for entire messages. The latency of a message from one network adapter to another depends on the number of links, their *bandwidth* and the current traffic through them.

The lower part of figure 3 depicts the `netsim` module, which exemplifies a network topology. We can see how peers with a good communication among them are grouped into a cluster and have individual links to the same ISP. In this example, peers `P1` and `P2` are connected at conceptual level, which at network level is achieved by connecting their corresponding network terminations `p1` and `p2` to the same ISP (`r1`). We also have the agent `A1`, which is the assistant of these peers, connected to the same ISP (`r1`) through its corresponding network termination `a1`. Each cluster is connected to the others by means of links, so `r1` and `r2` have aggregated links connected to `r0`, which represents the interconnection through Internet.

The simulator also includes an `overepast` module that processes the generated logs and extracts relevant information. This information can be later on displayed in different types of graphics. Hence, this can be used to compare the time spent to share the data in different configurations, or by using different sharing methods.

4.2 Sharing methods

Our simulator offers alternate sharing methods so that they can be executed over the same initial configurations in order to compare their results. Current available methods are: a single-piece version of the BitTorrent protocol (BT), the 2-LAMA approach with *social structure* adaptation but no *norm* adaptation and the 2-LAMA approach with *social structure* and *norm* adaptation.

Since the BitTorrent protocol inspired our 2-LAMA P2P approach, both protocols at peers' level are really similar (in fact, both protocols work with single-piece data and share most messages). Latency phase is not present in BT and our whole ML collapses into a single agent (*tracker*) that informs about all existing peers. As a consequence, the social structure phase is reduced to the tracker informing about all connected peers, ant thus, peers do not receive any further assistance to share the datum. Data sharing phase follows the algorithm described in [10]. In brief, it uses the same messages but decisions do not depend on norms but on protocol pre-fixed variables, so agents do not have any chance to take their own decisions.

In contrast, agents in the 2-LAMA approach can decide their actions as far as they respect norms. When executing the 2-LAMA approach *social structure* adaptation but no *norm* adaptation, *norm* parameters ($\texttt{max}_{\texttt{BW}}$, $\texttt{max}_{\texttt{Friends}}$, $\texttt{max}_{\texttt{Has}}$)

are fixed from start. On the contrary, the 2-LAMA approach with *social structure* and *norm* adaptation also has the *norm* parameters, but $\texttt{max}_{\texttt{BW}}$ and $\texttt{max}_{\texttt{Friends}}$ are self-updated at run-time at certain adaptation intervals ($\texttt{adapt}_{\texttt{interv}}$, an additional parameter). Each assistant computes their desired values for each norm taking into account the information collected from its cluster and the information received from other assistants. Assistants use a voting scheme as a group decision mechanism to choose the actual norm updates before notifying their peers.

4.3 Graphical User Interface (GUI)

By its very nature, MAS are complex systems composed by many agents acting and interacting simultaneously. Runtime monitoring information is usually low level, so we need graphical means to analyse system's evolution from a higher level of abstraction. With this aim, our simulator extends the Repast GUI and creates and advanced user-friendly GUI. Next sections detail its architecture and the new functionalities it provides.

Architecture Figure 4 shows the two main parts of the simulator: the core and the GUI. The core is based in Repast Simphony, which provides general simulation utilities such as schedulers or basic extendable models. By extending it, we have created our 2-LAMA P2P Simulator core, which implements domain-specific simulations. As for the GUI, our advanced GUI also extends Repast original GUI, providing some additional functionalities. It is able to exchange information with it and uses its basic tools to draw elements in the screen, such as the agents in the simulation, the messages sent among them, etc.

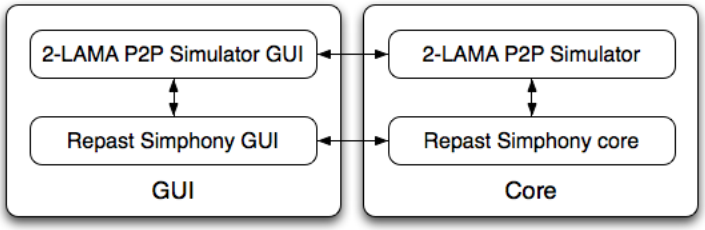


Fig. 4. Repast-based architecture of the 2-LAMA Simulator GUI

Our GUI also captures information from the P2P simulator in order to obtain all the information that will be displayed subsequently. This information comes down to the messages that are being sent among peers. The GUI is constantly listening to the simulator to catch these messages, which are stored afterwards. Figure 5 depicts the organisation of these messages. Peers are grouped in pairs, and these pairs are labelled with the name of its peers alphabetically ordered. Each pair of peers has a bag with the messages that one peer is sending to the

other at a given time step. For example, left side of the figure shows a group labelled `P1P2`. This group has a bag storing three messages from peers `P2` and `P1`, and the first one is a `PIECE` message that is being sent from `P2` to `P1`. The GUI uses this information to paint coloured arrows that represent the type and direction of messages that are being sent among peers.

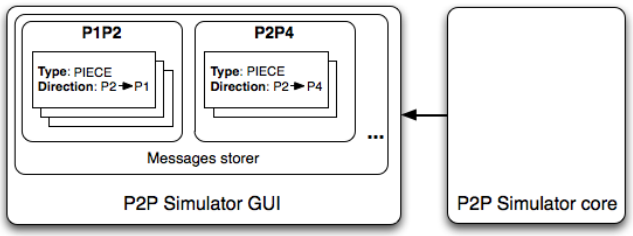


Fig. 5. Organisation of the information needed by the GUI to display simulation events

Functionalities This section explains the functionalities provided by our advanced GUI. Figure 6 depicts a screenshot of our simulator that illustrates GUI's general appearance. As for any other GUI, it has the natural aim of supporting user's interaction and the general objective of providing relevant information about the simulation —such as the messages sent at a given time step, their type, the source and target peers of the message, etc.

As Figure 6 shows, GUI functionalities are distributed in the following six main layout areas:

1. **Control toolbar**: This toolbar pertains to the original Repast GUI (and thus, it does not requires further implementation). It allows users to play the simulation, pause it or execute it step by step, where each step corresponds to a *tick* of the simulation.
2. **Legend of agents**: It shows how the different types of agents and possible states are displayed in the layout. Thus, the user can identify each agent and know if it is acting as an assistant or a peer, and have it into account to interpret what is happening in the simulation at every moment.
3. **Legend of message types**: It shows the colours corresponding to each type of message agents can exchange. This colour can be changed by means of the element on the left of the coloured line next to each message type. These changes can be saved into a file to recover them in future simulations.
4. ***Visible* and *Pause* checkboxes**: For each type of message, there are also two checkboxes that allow the user to show/hide the messages of that type that are exchanged among agents, or pause the simulation when any agent sends a message of that type.
5. **Runtime P2P Network layout**: This layout shows an animation of the agents of the simulation and the communications among them. Peers and

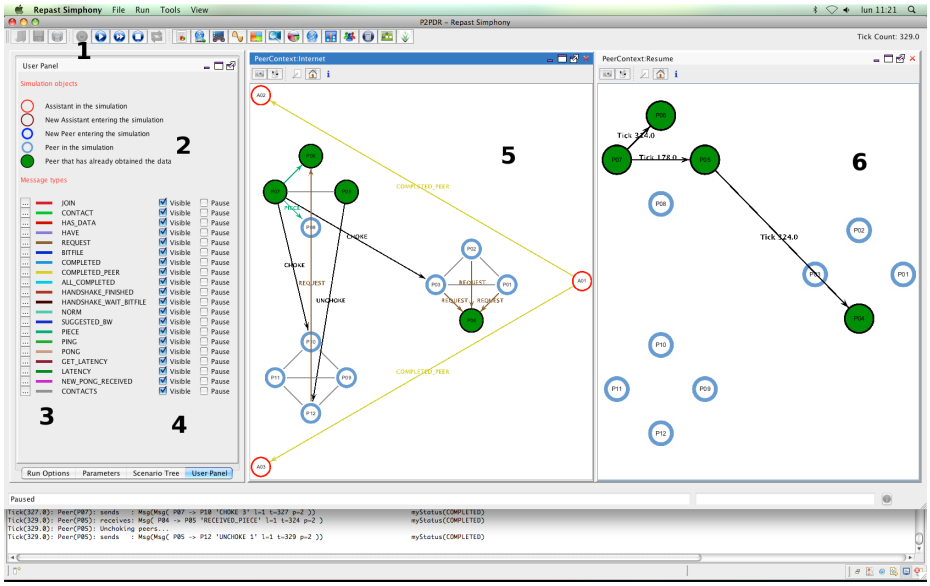


Fig. 6. 2-LAMA P2P Simulator Graphic User Interface

assistants are drawn according to the network topology. Following the example of Figure 6, peers are grouped in clusters of four peers, where each cluster is linked to an assistant. Messages sent among agents are displayed according to the defined colour in the user panel.

6. **Resume layout**: This layout shows how the data has been distributed among the P2P agent community. It also highlights completed peers and displays arrows connecting source and receiver agents. Furthermore, these arrows are labelled to specify at what time step the data was received.

Both *Runtime P2P Network* layout and *Resume* layout are able to draw new agents entering the simulation, highlighting them during some *ticks* and redistributing the layout to place them into the network.

4.4 Results analysis facilities

As it has been previously mentioned, our `overepast` module collects textual information (logs) and analyses it off-line (i.e., at the end of one or several executions). Analysis is done by generating additional plots that show how the main metrics change along execution time or with different simulation parameters. Thus, summarising and comparing the performance of different simulations. This turns out to be very useful for system designers, since rather than just knowing the overall system performance, it helps to understand its evolution based on detailed information such as bandwidth usage or link saturation. As a consequence, if some

problems arise, it is easier to identify them, their possible causes and what is most valuable: which simulation parameter values perform best. For a detailed analysis of results comparing the BitTorrent protocol and our 2-LAMA approach reader s referred to [11].

Specifically, the following metrics are graphically displayed: (1) time required to share the data; (2) network consumed; (3) mean number of hops that traveling messages perform; (4) network channel usage; (5) network channel saturation; (6) number of cancelled messages; (7) cost associated to these cancels; and (8) source factor —measure that provides information about how data was actually distributed among peers. Next figure 7 shows time performance comparisons for different sharing methods (with and without norm adaptation) and different norm values ($\texttt{max}_{\texttt{BW}}$ in $normBW_{DL}$ and $\texttt{max}_{\texttt{Friends}}$ from $normFriends_{DL}$) [6].

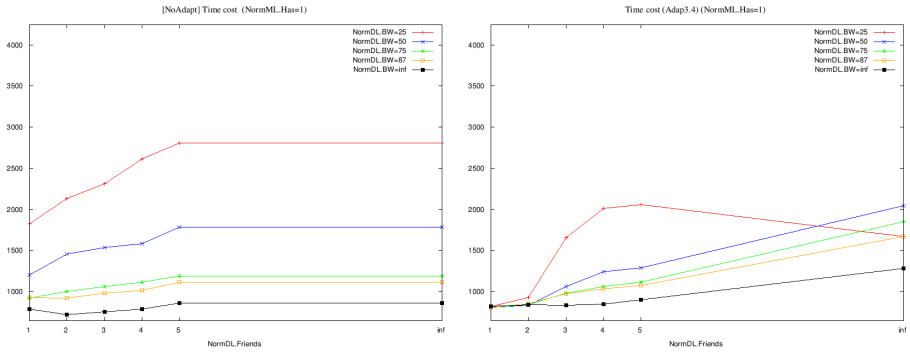


Fig. 7. Off-line comparison of different simulations

5 Conclusions and Future Work

This paper presents a simulator that has been developed with the aim of studying MAS organisational adaptation mechanisms in P2P scenarios. In order to endow the system with self-adaptation capabilities we advocate for adding a meta-level in charge of that task, instead of expecting participating agents to increase their behaviour complexity. The presented simulator provides the following functionalities and facilities:

- Definition and execution of simulations with different characteristics, as for instance network topology, number of peers. or sharing method.
- A GUI that permits to graphically control and follow simulations' evolution. Graphical representations are far more intuitive than textual logs, and if we

[6] In these plots $\texttt{max}_{\texttt{Has}}$ from $normHas_{ML}$ has been fixed to 1

also add the option to choose what to display (as in the user panel in our simulator), then the gain is even larger.
- Testing of different adaptation mechanisms for the social structure and norms.
- Process and analysis of simulation results. It generates different plots that help to compare the results of different simulations

Notice that while social adaptation is performed individually by each assistant within its cluster, in norm adaptation assistants have to reach an agreement on the norm new value. Specifically, each assistant computes its new desired norm values and later on they have to reach an agreement on each norm value. We believe that agreement technologies can play a key role in this process. Hence, our simulator can be used to test different approaches for reaching agreements among autonomous agents in the context of norm adaptation.

As future work, we plan to evaluate our approach with populations with norm violators, and in simulations where participants enter and leave. We are also interested in applying learning techniques to adaptation services.

Acknowledgements: This work is partially funded by IEA (TIN2006-15662-C02-01) and AT (CONSOLIDER CSD2007-0022) projects, EU-FEDER funds, the Catalan Gov. (Grant 2005-SGR-00093) and Marc Esteva's Ramon y Cajal contract.

References

1. Esteva, M.: Electronic Institutions: from specification to development. IIIA PhD. Vol. 19 (2003)
2. Hübner, J.F., Sichman, J.S., Boissier, O.: S-MOISE$^+$: A middleware for developing organised multi-agent systems. In: AAMAS Workshops. Volume 3913 of LNCS., Springer (2005) 64–78
3. Deloach, S.A., Oyenan, W.H., Matson, E.T.: A capabilities-based model for adaptive organizations. Autonomous Agents and Multi-Agent Systems **16**(1) (2008) 13–56
4. R., K., N., G., N., J.: Decentralised structural adaptation in agent organisations. In: AAMAS Workshop on Organised Adaptation in MAS. (2008)
5. Sims, M., Corkill, D., Lesser, V.: Automated Organization Design for Multi-agent Systems. Autonomous Agents and Multi-Agent Systems **16**(2) (2008) 151–185
6. Zhang, C., Abdallah, S., Lesser, V.: MASPA: Multi-Agent Automated Supervisory Policy Adaptation. Technical Report 03 (2008)
7. Campos, J., López-Sánchez, M., Esteva, M.: Multi-Agent System adaptation in a Peer-to-Peer scenario. In: ACM SAC09 - Agreement Technologies. (2009) 735–739
8. Campos, J., López-Sánchez, M., Esteva, M.: Assistance layer, a step forward in Multi-Agent Systems Coordination Support. In: International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS). (2009) 1301–1302
9. Lewis, D.: Convention: A Philosophical Study. Harvard University Press (1969)
10. Cohen, B.: The BitTorrent Protocol Specification. http://www.bittorrent.org/beps/bep_0003.html
11. Campos, J., López-Sánchez, M., Esteva, M., Novo, A., Morales, J.: 2-LAMA Architecture vs. BitTorrent Protocol in a Peer-to-Peer Scenario. In: to appear in Twelfth Catalan Congress on Artificial Intelligence (CCIA09). (2009)

Agreement Technologies for Adaptive, Service-Oriented Multi-Agent Systems

J. Santiago Pérez[1], Carlos E. Cuesta[2], and Sascha Ossowski[1]

[1] Centre for Intelligent Information Technologies (CETINIA), and
[2] Kybele Research Group, Dept. Comp. Languages and Systems II
Rey Juan Carlos University,
28933 Móstoles (Madrid), Spain
{josesantiago.perez,carlos.cuesta,sascha.ossowski}@urjc.es

Abstract. Multi-Agent Systems (MAS) are increasingly popular in Artificial Intelligence (AI) to solve complex problems. They can be conceived flexible and able to adapt to different situations. However, these features are often compromised by the characteristics of the problem itself. On the other hand, MAS have not had a lot of success in the industry, probably due to a different development culture. To solve this, MAS techniques should be more accessible to the general public, and have a shorter learning curve. The proposed approach is to use service-oriented concepts, which are popular in industry, to simplify this step. Moreover, if this approach manifests also self-adaptive capabilities, it will fulfil the notion's original promise: to guarantee that the system is able to adapt to changing conditions of the problem. This work proposes a service-oriented framework, consisting on a supporting agent-oriented architecture, a development methodology for service-oriented MAS, and an infrastructure based on the concept of agreement, which makes it adaptive. The first section provides a brief introduction and summarizes the paper goals. This is followed by the description of the base architecture, designed to support the agreement structure. Next section discusses concepts about service layers and the role of organizations. After that, the service-oriented methodology as well as the agreement structure itself is presented. Finally, a real-world case study, in the domain of medical emergencies, is analyzed, some conclusions are drawn, and further lines of work are outlined.

Keywords: Multi-Agent Systems, Service-Oriented Architecture, agreement, coordination, adaptability.

1 Introduction

The concept of *agent* has evolved, and nowadays MAS are increasingly popular in AI as a generic approach to solve complex problems. Different development strategies have been proposed in order to make them flexible and able to adapt to different situations. However, these features are often compromised by the heterogeneity of components, the nature of problems themselves, or the dynamism in the environment. On the other hand, MAS have not had a lot of success in the industry [14][36], probably due to a different development culture. To solve this, MAS techniques should be more accessible to the general software community. The proposed approach is to use service-oriented concepts, which are popular in industry, to simplify this step. Moreover, if this approach demonstrates also self-adaptive capabilities, it

will fulfil MAS original promise: to guarantee that the system is able to adapt to changing conditions in the problem to solve.

Before dealing with adaptability, it is perhaps better to consider coordination as a previous concept. A well-known definition of "coordination" within the MAS field is taken from Organizational Science: "the management of dependencies" between organizational activities [27]. From a "micro" point of view (agent-centred) [35], coordination is understood as an *adaptation to the environment*. On the other hand, from an MAS-centred point of view, the consequences of coordination can be understood as a global influence. This can be a "shared" plan [30] or the combination of individual plans (a "multi-plan") [28]. In few words, when using MAS as a software solution, the problem of coordination is always present. In fact, when we have a self-organized agent structure, we can often consider this structure as *optimal*, because it would solve the coordination issues.

Some early steps in the direction of adaptability have been given by organization-oriented approaches. Obviously there are many other approaches, but this is one of the most interesting in our context: adaptive capabilities, using a MAS approach, seem to be most easily provided by organizations. These imply a number of additional questions: about the inner role of organizations in MAS and about the need to provide coordination for organizations to achieve adaptation. To answer to them, two additional concepts have to be defined; respectively, *services* of an organization and *agreements* between them. The former provides both a methodological basis for the approach, as well as a direct connection to SOA [26]. On the other side, the latter is a main topic of this paper, and it will be discussed in detail.

Globally, this paper pursuits three main goals, namely:

- To evolve the classic agent-oriented approach, from an originally closed MAS design into an *open* Service-Oriented ecosystem,
- To define the corresponding infrastructure and methodology to achieve this, using the notion of *organization* as the conceptual nexus, and
- To provide internal *coordination* by defining the *agreement*, conceived as an adaptive architecture-level construction, which would provide coordination as an emergent property, by containment.

This paper is organized as follows: second section describes the base architecture, designed to support the agreement structure. Next section discusses concepts about service layers and the role of organizations. Them, the service-oriented methodology as well as the agreement structure itself is presented. Finally, a real-world case study, in the domain of medical emergencies, is analyzed, some conclusions are drawn, and further lines of work are outlined.

2 A Base Architecture for Service-Oriented MAS

The architecture that gives support to the model has been defined both as an open MAS and also as a service-oriented, organization-centric, agent-based architecture. These two perspectives are not necessarily contradictory; they are not obviously compatible either.

For both descriptions to be true, the platform has to be capable of being observed at different levels and from different perspectives. This multi-level and multiple viewpoint nature must be specifically enabled by the technical architecture (see 2.2), as it must present several different notions as the key concept of the system. This requires an intertwining relationship which must be purposely provided by the infrastructure. As the platform is conceived as a distributed system, the middleware is the logical place to provide this support.

2.1 The Need for Organizations

As defined previously, the architecture that supports the model has been defined as open MAS, which is also service-oriented, organization-centric and, of course, agent-based. In this work, agents supporting services has been chosen as the solution alternative. First, agents are well-known computational entities in the academic environment, with an implied granularity, and need to comply with an existing standard [20]. On the other hand, although the services technology is established and has a number of standards [7][12][17][26], its methodology and influence on other paradigms are still under development. In order to allow the use of the rich semantic and technological capabilities of agents in a broader context, an upper layer of services can be added to provide, in particular, the *interoperability* feature. Therefore, it is easy to conceive a service as a way to present the operational capabilities of an agent or, even better, a collection of agents as an organization. One way to implement is to have the platform defined as a SOA, built on top of supporting MAS.

Implicit in the definition of MAS is the need to *register* agents in the system, to separate those ones who belong to the architecture from those who do not. The same approach will be used to identify services. To allow their external access, they will be explicitly registered and grouped as part of a service. This service could be later discovered by other entities within the distributed registry of the system.

Pure *agent-oriented* MAS methodologies (such as MAS-CommonKADS [24], Gaia [38], MaSE [37], Tropos [23] or Prometheus [29], among others) usually concentrate in the agent vision. It is assumed that the final behaviour of the system *emerges* from the interrelations between the designed agents. But the global behaviour is not analyzed in detail.

On the other hand, in *organization-oriented* MAS methodologies, the analysis is made from a global perspective (Agent-Group-Role [19], MESSAGE [9], ANEMONA [22], AML [11], OperA [15], Civil Agent Societies [13], MOISE [21], Electronic Institutions [18], HARMONIA [34], GORMAS [3], among others). The objectives describe the organizational purposes at a high level. This allows the determination of tasks, types of agents, resources assignation between members, etc. In this approach, norms are very important because they describe the desired behaviour of the members. These norms will derive in control, prohibitions, sanctions, etc. to achieve the expected global behaviour. Mechanisms to allow external agents to enter the organization and control their behaviour are particularly useful to design *open* MAS.

2.2 The Agreement Technologies Base Architecture

The set of technologies and approaches used in this work is globally named as “Agreement Technologies” [1]. This section presents the base architecture for these technologies, and, as it was noted in the previous section, it was conceived to be based in an open MAS.

One goal of the proposed approach is to take advantage of MAS features, so the research is oriented to achieve a greater capacity and functionality, with a lesser emphasis on efficiency or scalability. Moreover, and from this point of view, services are used to achieve interoperability, as mentioned earlier. The main idea is to export the agent system as a system of services, which will be supported, not only technologically, but also methodologically.

These concepts are intended to be built on top of existing and concurrent work. It is not the purpose of the article to give a complete description of the THOMAS architecture, which can be found in [4]. But briefly, its design can be summarized as described in the following.

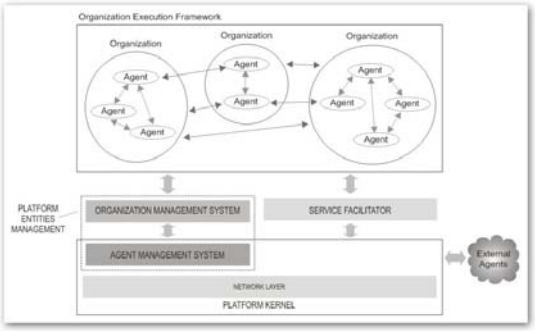


Figure 1: THOMAS Technical Architecture (inspired on [4])

The platform, including its middleware, (Figure 1) is structured in three levels but they are not strictly layers. They are orthogonally supported by four specific components, which are included as part of three different subsystems. The *Platform Entities Management* subsystem is actually layered in turn. The different layers of this subsystem are used to provide capabilities for different levels in the platform. The three levels are:

- *Platform Kernel (PK)*. It is the actual kernel of the middleware; includes both the Network Layer and the Agent Management System (AMS) component. It provides all the capabilities of FIPA-compliant architecture [20]. Therefore, at this layer the platform is already an (open) Multi-Agent System.
- *Service & Organization Management*. This is the conceptual level composed of the Organization Management System (OMS) and the Service Facilitator (SF) components. Both components provide all the relevant features and abstractions for the Execution Framework.
- *Organization Execution Framework*. It is the "space" where all the computational entities "live" and perform their functions. Agents and their organizations, and the services they offer, are conceptually located in it. Every specific application would be conceived, designed and executed at this abstraction level.

The aforementioned three main components of the platform are: *AMS*, which provides all the required capabilities and functions for managing an agent; *OMS*, which provides all the required capabilities and functions for managing an organization, and maintains together the system as a whole; and *SF*, which provides the required capabilities and functions to allow that a certain selection of the operations in an organization behave as a unified service.

3 The Service-Oriented Layer

As already noted, the base architecture will be primarily conceived as a service-oriented. Hence, an important concept is that of *service*.

According to their provider, there are basically *base services* (user-level services, and they are defined for every concrete application); and s*ystem services* (not strictly "services" as they are not offered by a concrete user-level provider, they are provided by the system itself, i.e. they are the support services of the platform).

Taken into account their function and the extent of their capabilities three separate sets of services can be identified in the architecture:

- *Structural Services*. They allow defining a certain organizational/architectural structure, by creating and registering organizations, their roles and norms, and their relationships. They make possible to establish and modify both structural and normative specifications of the system and they are provided by the OMS.
- *Information Services*. They provide specific information about components in an organization. Also, some of them are published as registered services, while some others are just conceived for the use of the OMS and stay invisible.
- *Dynamic Services*. They allow entities to dynamically enter or abandon an organization, as well as to adopt existing roles. Units and roles have been previously defined and registered by using structural services. Dynamic services are just able to modify services, units and roles. These services provide dynamic reconfiguration.

3.1 The Role of Organizations

The *organization* is the most important active element and the unifying notion of the architecture itself. The recursive hierarchy of organizations is what would make possible to simultaneously define the architecture as service-oriented and as agent-based. The concept of organization is the nexus between both perspectives.

An organization can be seen from two points of view: externally, it can be considered as a context, a domain of influence, the scope of a set of norms and rules; and internally, it can be considered as a collection, the gathering of the set of individuals which would comply with the stated norms and fill the defined roles. An organization is also composed of units (or organizational units). A *unit* is an active entity with a definite, externally observable behaviour, and it can have either a collective nature (where the unit is itself an organization) or an autonomous nature (when the unit is just a single agent). The *unit* is therefore the substrate which supports both the gathering of agents and the definition of services.

The concept of organization is also used to solve the scaling problem of architecture, in the context of services. Since they generally are intended to be used in-the-large, it is necessary to use a compositional structure: the organization itself. In this vision, low-level services are essentially provided by individual agents, while system-level services are provided by roles in a complex organization. Intermediate levels can also provide their services, so the recursive organizational hierarchy defines the compositional “spine” for the system.

As implied before, from this point of view everything is a *unit*. The system itself must be conceived from within as a unit, and therefore, it is an organization too. As such, it gathers the contributions of both individual agents defining the small-scale MAS, as well as those from the middleware itself, which supports the technical architecture, as described in section 2.

4 A Service-Oriented Methodology

As already said, the proposed approach is to group agents into organizations, but this is not a simple task. Some questions arise, such as: Which agents belong to an organization? What criteria will be used to group them? Moreover, the process of exporting the capabilities of agents as services leads to another question: What services should be exported?

A methodology is proposed in an attempt to answer all these questions. A first step involves the functional decomposition of services, and this leads to define organizations. Then, as a second step, the composition of services is guided by the organizations and their structure.

The system is conceived as service-oriented, so, high-level services are proposed as the starting point. Their functional decomposition (or a hierarchical decomposition, from another point of view) will be also used to design the hierarchical structure of organizations.

A service is defined as a computational entity which gathers a set of operations, described in its standard interface, and comprised a semi-ordered sequence of activities, semantically described by an intentional profile and an explicit process model, which can in turn be split in several smaller processes. There may be several implementations for the same service and an identical profile, which are offered by different (possibly many) service providers.

The concept of *service process*, in this context, intends to provide a clear semantic perspective of a service's functionality, by describing it as a workflow.

The service process model identifies three kinds of processes in the structural description. This classification, designed from a semantic perspective [2], will be used to support the methodology, and assist in the design of the structure of organizations. These types of processes are:

- *Atomic processes* can be directly invoked, execute in a single step, and cannot be decomposed.
- *Simple processes* are also perceived to be executed in a single step, but cannot be directly invoked. They are abstract processes (placeholders) and can be filled either by an atomic process; or (acting as a simplified representation) by a composite process.
- *Composite processes* are decomposed in sub-processes, which can be defined in turn as atomic, simple or composite ones. This way, the service's functionality unfolds recursively as a hierarchic composite structure.

Simple processes (which are also services) allow a next level of decomposition. High-level services can be described as a set of simple processes. Those actually simple are described as *atomic* services (i.e. agent operations); and those that are more complex are considered as *composite* processes, which will be further decomposed. Organizations can be now identified by relating each service with its provider, unfolding their hierarchical structure.

From this point of view, the composition of services is given by the organizational structure itself. Though the approach here has a semantic nature, this is essentially the same approach which is also used for this purpose, from a behavioural perspective, in the context of service composition, based on orchestration [25].

In particular, both approaches use the process abstraction as the way to describe the behaviour of a service, and specifically the composition of (smaller-scale) services. Also, provide a number of control structures, which define a principled way to combine sub-processes into larger processes, providing compositionality and recursive structures.

There is an implicit relationship between these recursive structures: (composite) processes can be provided as services by (composite) organizational units; when these processes are decomposed, the resulting sub-processes can be provided in turn by other units. That is, sub-processes of a composite process would be provided by the members (units) of the composite organization which provided the upper level. When this happens, the recursive structure of processes mimics the recursive structure of organizations. The converse is also true: starting from simple tasks, a vertical composition method could help in the definition of the organizational hierarchy, defining at the same time the resulting complex (composite) processes. Like in the case of organizations, the recursion ends at the agent level.

Therefore, our approach provides the structure for the vertical composition of services. This way, a task that is often considered difficult –to design the service composition– is methodologically tackled, allowing at the same time to fully exploit the organizational structure of the agents. Then, there is a mutual support between these two concepts.

5 The Agreement Structure

Agents were originally conceived as single actors, but within the MAS approach, a different method has become possible. The need for a trade-off continues, but has it transformed into a *coordination* problem. As already said, the service can be conceived as a way to present the operational capabilities of an agent (or a collection of them) inside an organization.

The proposed methodology allows tackling the decomposition of services, but adaptability in the system is provided by the architecture. First, there is a decomposition of services to provide the required features; but after that it is necessary to address the structure of agreements which supports this decomposition, in order to make it adaptive.

So, an important notion is the *agreement* between computational entities (organizations, at the top levels; but also agents, at the lower ones) conceived as an *architectural construct*. The following subsections discuss the need for an adaptive structure, and the agreement model.

5.1 The Need for an Adaptive Structure

When using MAS as a software solution, as already noted, the problem of coordination is always present. When they define a self-organized structure, it sometimes implicitly solves the coordination issues; this approach could be considered as optimal.

When a complex problem is tackled in an *ecosystem* (or a system of systems), the solution requires certain adaptability. At the same time, this structure needs to be flexible to achieve coordination inside the ecosystem, and also this behaviour could be emergent.

Pioneer works related to cooperation define adaptiveness as a required notion for intelligent solution of complex problems [2]. Two approaches can be considered: *from the collaborative entity point of view* (cooperation is introduced as an additional mechanism to increase the effectiveness in solving problems); and *from the problem to solve point of view* (this intends to find the best way to structure and decompose a complex problem to solve it effectively). Taking into account these approaches, several solutions to the cooperation problem were developed. The *blackboard* architecture [16] provides cooperation between knowledge sources using a simple communication mechanism. The *contract net* [32] proposes *negotiation* as a mechanism to coordinate and to assign tasks to different entities participating in problem solving. The *reactive architecture* [8] tries to obtain an intelligent behaviour from simple models, without knowledge representation, reasoning or learning mechanisms. Finally, agent architectures with *organizational* capacity appeared: agents need to know about their own capabilities and *social* features.

Generically, entities are organized into a structure by using *controls*, which either *enforce* or *forbid* specific interactions –or connections–; and *protocols*, which either *enable* or *channel* them. Therefore, where the former are based on force or *imposition*, the latter are based on consensus and *agreement.*

The concept of agreement among computational entities seems to be a right approach to tackle the need for an adaptive structure. The objective is to “discover” a suitable structure of controls and protocols so that it *emerges* as a global structure, *the agreement*. This will make possible to define the main inner structures in order to obtain agreement-based organizations.

As the structures of agents are become more and more complex, it is clear that for some kind of problems we need not a superstructure, like the blackboard. Agents that *organize themselves* in organizations (and after that in agreement-based organizations) are needed. The main objective is to evolve from that emergent coordination to an *emergent agreement* between entities.

5.2 The Agreement Model

As already noted in previous sections, a central notion in this approach is the *agreement* between computational entities. Continuing with research efforts in the field of "Agreement Technologies" [1], the process of *agreement-based coordination* can be conceived as consistent with the normative context where agents are established and allow them, once accepted, to call for mutual services, and to be called by others.

Several key research topics must be considered and they can be seen in a "tower" structure [1] where each level provides functionality and inputs to the one above. Therefore, the agreement must be seen as a *layered* structure, by definition: when an agreement is reached at a certain level, elements located at lower levels must respect it at their own level. These "tower" levels, from bottom up, are:

- *Semantics*: the bottom one, as semantic issues influence all others. The semantic alignment of ontologies [6] is necessary to avoid mismatches and is needed to have a common understanding.
- *Norms*: is concerned with the definition of rules determining constraints that the agreements, and the process to reach them, have to satisfy.
- *Organizations*: implies a super-structure that restricts the way agreements are reached by fixing the social structure of the agents, the capabilities of their roles and the relationship among them [5].
- *Argumentation and Negotiation*: can be seen as protocols that define the structure of an agreement.
- *Trust*: the top level in the tower. Agents need to use trust mechanisms that summarize the history of agreements and subsequent agreements executions in order to build long-term relationships between them [31].

These five layers, of course, are not seen as isolated because they may well benefit from each other. For example, if changes in some norms or to take advantage of negotiation methods, the organizational model has to be modified. A switch from the described "tower" into a multi faceted ("pentagon") figure can be conceived because the agreement pervades (and is influenced by) all the facets/levels (Figure 2). In this sense, the facets are intertwined, but *agreement* is still a layered structure – and layers bind both ways.

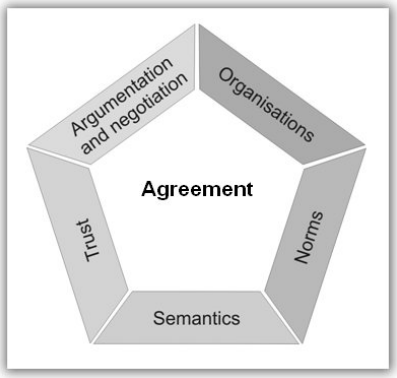


Figure 2: Multi-faceted perspective on the structure of an Agreement

In fact, the agreement is a crosscutting structure, which maintains a bidirectional relationship to every element it contains. The agreement defines the architecture but at the same time, the architecture defines the agreement. The agreement is *shaped* by those forces,

but its existence also *shapes* the reaction to them, and models the future evolution of the system. It is important to note that the multi-faceted perspective is not intended to replace the "tower" structure, as the architecture described in previous subsection is still hierarchical in many senses, but the *agreement* itself can be considered not only as layered, but also as multi-faceted. Layers are just conceived to provide logical separation of concerns, and they are not always physical (contained) tiers. On the other side, in an MAS setting, a reconfiguration can also be triggered bottom-up; a single agent can react to a change in their surroundings by asking for some kind of change, such as a *move* to some other organization. Of course this change can cause some others in turn, and the effect would spread accordingly, causing even a global reorganization.

In summary, the system already provides the required elements to build an adaptive architecture; to actually define an *emergent agreement* would just require identifying the structural patterns, and the set of inter-level protocols. Some refinements can be made further, though the need for meta-elements has still to be considered, nothing excludes the definition of specific agents to carry out support tasks for the agreement itself (such as sensors, observers or even planners).

6 Case Study: *mHealth*

This section presents a case study in order to illustrate the proposed approach. Our purpose is to show the reason why an *agreement* between entities is not only necessary, but it can also be a natural solution to complex problems. As already said in section 3.1, the structure in organizations can be seen as a logical strategy to tackle complex situations, and has also several advantages. The need for a flexible and adaptive agreement construct can also be seen as the basis to create and evolve these organizations. Section 5 has described the structure of the *agreement* structure which could address that need.

The example is related to the *mHealth* (mobile-Health) demonstrator, which is an evolutionary prototype currently under development within the Agreement Technologies project [1]. It is inspired by work with SUMMA112 [33], the centre that manage medical emergencies in the Autonomous Region of Madrid, which is also involved in the project.

In the following, an initial emergency (E1) is described. The system has to evolve to simultaneously react to a second one (E2).

E1. There is a fire in Casa de Campo (a large urban park). There are about 500 people at that moment and about 65 of them present symptoms of asphyxia. *SUMMA112* receives information related to E1 and decides that 5 ambulances and one helicopter are needed. The coordination with hospitals near the area, Fire Department (*FD*) and Police (*P*) is also urgent. *FD* and *P* will send 3 fire trucks and 5 police cars. From an organizational approach, all these elements form an organization, **O1**. Each actor maps onto an agent considering this scenario as MAS. Then, there are 14 agents are interacting in the organization O1. Each agent has its role, goals and plans inside the organization, which in turn has its own norms and protocols.

E2. One hour after E1, there is a chain car crash (E2) in the tunnel of Paseo de Extremadura, a road near to E1 location. Several cars have crashed and 2 of them are on fire. *SUMMA112* decides that this emergency requires 3 ambulances. In this case, *FD* and *P* decide to send one fire truck and 3 police cars. Again, all these 7 elements form a second organization, **O2**.

Basically, this scenario can be solved using two alternative solutions: deal with O1 and O2 as separate elements, with no relation between them; or, deal with O1 and O2 as *units with some degree of relationship*.

The second is the most efficient and sensible approach, as it must have into account potential interactions between both emergencies. So, let's consider first O1, whose elements reach an *agreement* to tackle E1. At this point, the agreement construct can be seen as "the set of elements interacting in a coordinated way to solve a problem". But at the time to assign resources to E2, O2 is not considered in isolation from O1. Some resources that previously were mapped onto O1 now can be mapped on O2 because the conditions in emergency E1 may have changed during the last hour. This process of re-mapping implies a *reconfiguration* of unit O1, i.e. an agent's *reorganization* within the *O1O2 composite*.

Some services which were provided by unit O1 are no longer required in E1 and now can be re-mapped onto O2. This can be done at different levels (for instance, registering services at the unit level, with no structural changes); but the simplest and most efficient solution implies not only re-assigning *services*, but also the *agents* which provide them, i.e. doing a *reorganization*. For example, according to the observed results in O1 some services can be assigned to E2. Additional elements are also assigned to E2 to fulfil O2 necessities. O1, a smaller unit now, continues working in E1; and a new agreement is created around E2, defining the O2 organization. At the same time, a larger *agreement* is created encompassing both units (and therefore, defining another one). This agreement would continue adapting to changes in both emergencies as system evolves.

Elements participating in an *agreement* (*O1+O2*) must be capable to adjust themselves to environmental changes, to accomplish the goals in the agreement. This will often lead to changes, not in the elements themselves, but on their configuration. In fact, even the criteria used to decide if an agent belongs in an agreement should be managed the same way: this defines an *emergent agreement*, where not only part of the behaviour, but the structure itself emerges from the situation.

The base architecture described in Section 2 already includes all the services and facilities necessary to carry out any reconfiguration [4]. However, this is not enough to define a self-adaptive structure – the triggering of those services is essential. Of course *norms* (to define constraints) and *organizations* (to define their scope) can assist in the establishment of such a structure; and even the *negotiation* layer can be used to trigger the creation of the agreement itself.

7 Conclusion

It has been argued that MAS techniques should be more accessible to software community in general in this paper. As services are concepts very popular in industry and can simplify the transition, this work has proposed a service-oriented framework, consisting on a supporting agent-oriented architecture; a development methodology for service-oriented MAS; and an infrastructure based on the concept of agreement, which makes it adaptive.

The example shows why it is needed to consider a general *ecosystem*, instead a "classic" *closed* system or a single-design *open* system. To actually provide the required response in an emergency, SUMMA112 has to coordinate with the information systems from the Fire Department, the Police, and every hospital in the area. This implies that it is not possible to have a unified pre-*programmed* strategy to manage emergencies, as it should be embedded in several independent systems which only sometimes gather to act together.

The key idea in the Agreement Model is that it creates an architectural *context*, in which agents (organizations, services) are coordinated and reorganized by inclusion in a structure. In particular, there is *not* an architectural element in charge of reconfiguration, i.e. there is not a

self-*supervisor*. Instead of that, every self-property in the system is conceived as *emergent*, and they will be "indirectly" provided by structural features of the agreement. The elements do just what they must to comply with the requirements of the location they occupy within the architecture; the relationships between the *agreement facets* will do the rest. Again, the case study discussed previously describes a simulated coordination effort in the current SUMMA112 system. In [10], MAS structured in organizations, and implemented in THOMAS architecture has been used to model systems and simulate several situations.

The reconfiguration process has also been modelled and tested using several different approaches; but this manual process is only the first stage of research. The next step is to develop a *model-driven* approach to guide the reconfiguration, and will be followed by a well-defined self-adaptive, emergent approach, which is the ultimate goal.

Acknowledgment: This work has been partially funded by Project AT (CONSOLIDER CSD2007-0022, INGENIO 2010) of the Spanish Ministry of Science and Innovation, and from COST Action AT (COST IC0801) from the EU RTD Framework Programme.

References

[1] Agreement Technologies (AT) Project: http://www.agreement-technologies.org/ (2009)
[2] Ana Mas: Agentes Software and Sistemas Multi-Agente: Conceptos, Arquitecturas y Aplicaciones. Prentice-Hall (2005).
[3] Argente, E.: GORMAS: Guidelines for ORganization-based Multiagent Systems. PhD thesis, Universidad Politécnica de Valencia (2008).
[4] Argente, E., Botti, V., Carrascosa, C., Giret, A., Julian, V., and Rebollo, M.: An Abstract Architecture for Virtual Organizations: The THOMAS Project. Technical report, DSIC, Universidad Politécnica de Valencia (2008).
[5] Argente, E., Julian, V., and Botti, V.: Multi-Agent System Development based on Organizations. Electronic Notes in Theoretical Computer Science 150(3):55-71 (2006).
[6] Atienza, M., Schorlemmer, M.: I-SSA - Interaction-situated Semantic Alignment. Proc Int. Conf. on Cooperative Information Systems (CoopIS 2008) (2008).
[7] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orchard, D.: Web Services Architecture. W3C WSA Working Group, W3 Consortium (2004)
[8] Brooks, R.: Intelligence without Representation. Art. Intelligence, 47:139-159 (1991).
[9] Caire, G., Coulier, W., Garijo, F., Gomez, J., Pavon, J., Leal, F., Chainho, P., Kearney, P., Stark, J., Evans, R., and Massonet, P.: Agent-oriented analysis using MESSAGE /UML. LNCS vol. 2222:119–125 (2002).
[10] Centeno, R., Fagundes, M., Billhardt, H., and Ossowski, S.: Supporting Medical Emergencies by MAS. In "Agent and Multi-Agent Systems: Technologies and Applications". LNCS, vol. 5559:823-833. Springer (2009).
[11] Cervenka, R., and Trencansky, I.: AML. The Agent Modeling Language. Whitestein Series in Software Agent Technologies and Autonomic Computing. Birkauser (2007).
[12] Christensen, E., Curbera, F., Meredith, G. and Weerawarana, S.: Web Services Description Language (WSDL) 1.1. W3C Consortium. W3C Note (2001)
[13] Dellarocas, C., and Klein, M.: Civil agent societies: Tools for inventing open agent-mediated electronic marketplaces. In ACM Conf. Electronic Commerce (EC-99) (1999).
[14] DeLoach, S.: Moving multi-agent systems from research to practice. International Journal of Agent-Oriented Software Engineering - Vol. 3, No.4 pages 378 – 382 (2009)
[15] Dignum, V.: A Model for Organizational Interaction: Based on Agents, Founded in Logic. PhD thesis, Utrecht University.
[16] Erman, L., Hayes-Roth, F., Lesser, V., Reddy, R.: The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty. ACM Computing Surveys 12(2), pages 213-253 (1980)

[17] Esteban, J., Laskey, K., McCabe, F., and Thornton, D.: Reference Architecture for Service Oriented Architecture 1.0. Organization for the Advancement of Structured Information Standards (OASIS) (2008).
[18] Esteva, M., Rodriguez, J., Sierra, C., Garcia, P., and Arcos, J.: On the Formal Specification of Electronic Institutions. Agent Mediated Electronic Commerce 1991, pages 126–147 (2001)
[19] Ferber, J., Gutkenecht, O., and Michel, F.: From Agents to Organizations: an Organizational View of Multi-Agent Systems. In Proc. AAMAS03 - Agent-Oriented Software Engineering Workshop (AOSE) (2003).
[20] FIPA. FIPA Abstract Architecture Specification. Technical Report SC00001L, Foundation for Intelligent Physical Agents. FIPA TC Architecture (2002).
[21] Gateau, B., Boissier, O., Khadraoui, D., and Dubois, E.: MOISE-Inst: An Organizational model for specifying rights and duties of autonomous agents. In der Torre, L. V., and Boella, G., eds., First Intl. Workshop on Coordination and Organisation (2005).
[22] Giret, A.: ANEMONA: Una metodología multi-agente para sistemas holónicos de fabricación. PhD thesis, Universidad Politécnica de Valencia (2005).
[23] Giunchiglia, F., Mylopoulos, J., and Perini, A.: The Tropos Software Development Methodology: Processes, Models and Diagrams. In Proc. Workshop on Agent Oriented Software Engineering (AOSE), 63–74 (2002).
[24] Iglesias, A., Garijo, M., Gonzalez, J., and Velasco, J.: A methodological proposal for multiagent systems development extending CommonKADS. In Proc. 10th Banff Workshop Knowledge Acquisition for Knowledge-Based Systems (1996).
[25] Jordan, D., Evdemon, J., Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., Guizar, A., Kartha, N., Kevin Liu, C., Khalaf, R., Koening, D., Marin, M., Mehta, V., Thatte, S., van der Rijn, D., Yendluiri, P., and Yiu, A.: Web Services Business Process Execution Language (WSBPEL) 2.0. Organization for the Advancement of Structured Information Standards (OASIS) (2007).
[26] MacKenzie, C., Laskey, K., McCabe, F., Brown, P., and Metz, R.: Reference Model for Service Oriented Architecture 1.0. Organization for the Advancement of Structured Information Standards (OASIS) (2006).
[27] Malone, T., Crowston, K.: The Interdisciplinary Study of Co-ordination. Computing Surveys 26 (1). ACM Press, pages 87–119 (1994).
[28] Ossowski, S.: Co-ordination in Artificial Agent Societies, LNAI 1535. Springer (1999).
[29] Padgham, L., and Winikoff, M.: Prometheus: A Methodology for Developing Intelligent Agents. In Proc. Agent Oriented Software Engineering (AOSE), 135–145 (2002).
[30] Rosenschein, J., and Zlotkin, G.: Rules of Encounter – Designing Conventions for Automated Negotiation among Computers. MIT Press (1994).
[31] Sierra, C., Debenham, J.: Information-Based Agency. Proc Intl. Joint Conference on AI (IJCAI-2007). AAAI Press, pages 1513-1518 (2007).
[32] Smith, R.: A Framework for Problem Solving in a Distributed Processing Environment. PhD thesis, Stanford University (1978).
[33] SUMMA112: http://www.madrid.org/cs/Satellite?language=es&pagename=SUMMA112%2FPage%2FS112_home (2009).
[34] Vazquez-Salceda, J., and Dignum, F.: Modelling Electronic Organizations. Lecture Notes in Artificial Intelligence 2691:584–593 (2003).
[35] Von Martial, F.: Co-ordinating Plans of Autonomous Agents. LNAI 610, Springer (1992)
[36] Weyns, D., Helleboogh, A., and Holvoet, T.: How to get multi-agent systems accepted in industry? International Journal of Agent-Oriented Software Engineering - Vol. 3, No.4 pages 383 – 390 (2009)
[37] Wood, M., DeLoach, S., and Sparkman, C.: Multiagent system engineering. Journal of Software Engineering and Knowledge Engineering 11:231–258 (2001).
[38] Wooldridge, M., Jennings, N., and Kinny, D.: The Gaia Methodology for Agent-Oriented Analysis and Design. J. Autonomous Agent and Multi-Agent Systems 3:285–312 (2000).

Developing Virtual Organizations Using MDD

Jorge Agüero, Miguel Rebollo, Carlos Carrascosa, Vicente Julián

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera S/N 46022 Valencia (Spain)
{jaguero, mrebollo, carrasco, vinglada}@dsic.upv.es

Abstract. Virtual Organizations are novel mechanisms where agents can demonstrate their social skills, due to the fact that they can work in a cooperative and collaborative way. Furthermore, organizations are frameworks where agents can achieve different types of agreements. But the development of organizations in MAS (Multi-Agent Systems) requires extensive experience in different methodologies and platforms. MDD (Model Driven Development) is a technique for generating application code developed from basic models and meta-models using a variety of automatic transformations. This paper presents a meta-model of *Virtual Organization* (of agents) using concepts and components at a suitable level of abstraction so that it can be implemented on different systems following a MDD approach. Based on this idea, a service-oriented organizations meta-model that is platform independent is presented. As an example, two model transformations that allow the unified model of the virtual organization to be translated into two different platforms are shown, facilitating the development process of agent-based software from the point of view of the user.

1 Introduction

Advances in new technologies based mainly on the Internet and the Web, such as electronic commerce, mobile/ubiquitous computing, social networks, etc., demonstrate the need to develop distributed applications with some intelligent capabilities. Multi-Agent Systems (MAS) are a powerful technology with very significant applications in distributed systems and artificial intelligence[18]. Supporting all of these developments requires the creation of platforms of highly heterogeneous agents, where agents work together through different interactions to support complex tasks, in a collaborative and dynamic way. Bearing this in mind, it is suitable for agents to display characteristics such as sociability, autonomy, self-organization, etc. Therefore, it is necessary to create open systems composed of a group of cooperative and heterogeneous agents, which work with local or individual goals and that must fulfill global goals. That is, a set of agents can reach agreements to achieve the group goals. The concepts of *organization* allow individual and group entities to be modelled in a very abstract way[3]. The

organization describes the main aspects of a society based on different viewpoints, such as: structure, functionality, norms, interactions, and environment. This type of organization is called *Virtual Organization*[13].

However, to implement an *Agents Virtual Organization* requires vast experience in one or more design platforms. A major challenge when designing MAS is to provide efficient tools that can be used by any user (non-expert users). The MDD approach can facilitate and simplify the design process and the quality of agent-based software[19], since it allows the reuse of software and transformation between models. MDD basically proposes the automatic generation of code from the models using the transformations. In other words, using models that have components that are platform independent, and by means of the transformations, those models are translated into components (or code) that depend on the execution platform, which integrates specific details about the system. Recently some proposals to implement these ideas (MDD techniques) have been proposed in MAS[16, 14, 17, 2], but none of these proposals focus on organization design.

Our purpose is to use the MDD approach for the design of organizations, taking the core and fundamental concepts, to specify an organization meta-model that is platform independent, and then use the translation mechanism to convert the unified meta-model into platform specific models for the execution of the agents. This paper presents a *Virtual Organization* meta-model, focused mainly on the integration of service-oriented technology and MAS, which allows support (from a very high level of abstraction) to be given to dynamic and open agent societies. Two transformation models for moving the unified model of the Virtual Organization to two different platforms are also proposed, allowing the feasibility of the proposal to be verified. These target platforms are: THOMAS[1] [7] and E-Institutions[2][12].

This paper is structured as follows. A brief summary of relevant works and their problems are discussed in Section 2. Section 3 presents MDD concepts and how to apply these concepts to the MAS paradigm. Section 4 explains how to design a virtual organization from the MDD viewpoint. Finally, the conclusions of this work are presented in Section 5.

2 Related Work

This section presents some related contributions with respect to organization modeling in agent-based systems, and discusses some problems. Furthermore, this section explains how, using the MDD approach, these problems can be addressed.

MAS development needs methodologies that allow the design of agent-based software to be optimized. The first methodologies that emerged can be classified as *agent-oriented*, without describing the *organization* explicitly. These systems are generally closed and external agents are prohibited. But in recent times, new *organizational-oriented* methodologies have emerged, which allow (partially) the

[1] http://users.dsic.upv.es/grupos/ia/sma/tools/Thomas
[2] http://e-institutions.iiia.csic.es

design of open MAS, leading to the development *heterogeneous* systems. These organizational-oriented methodologies allow external agents to access the system functionality, but the agents are obliged to adhere to the *social norms* of the system. Among the most important methodologies which allow the design of virtual organizations are: PASSI[9], MOISE[15], OperA[11] and GORMAS[4].

Now, with respect to the application of MDD for MAS, all of the above commented methodologies allow (to a major or minor extent) the design of agent organizations using their own set of specifications. But, in general, these proposals only use specific aspects of the platforms, which are in the majority of cases very different. This situation can create an added complexity for developers which try to employ these approaches. Moreover, some of them do not have an implementation phase, only defining high-level models, and difficulting enormously the developer work when tries to obtain executable code. But with the appearance of the MDD, it is possible to take advantage the flexibility provided by this approach to solve "some" of the problems found in the design of organizations in MAS.

The purpose of MDD is to create models legible by computers that can be understood by automatic tools to generate templates, proof models and even code, integrating them in multiple platforms[19]. From the viewpoint of the development of agent oriented systems, application development consists of obtaining the agent code that can be executed in different platforms. That is, to concentrate on the development of the application from a unified agent model and apply different transformations to get implementations for different platforms. Currently, the most common methodologies for MAS have an identified set of models that specify their characteristics. These models can be adjusted as MDD models that specify the concepts of the MAS as roles, behaviors, tasks, interactions or protocols. The models can be used to model a MAS without focusing on platform-specific details and requirements. After this, it is possible to transform any agent model into agent implementations for different platforms.

Only a few agent development methodologies have integrated the MDD techniques in the MAS design. The most relevant are MetaDIMA[16], TROPOS[6], PIM4AGENT[17], INGENIAS[14], AML [8] and AUML [5]. All of this work is for the application of MDD to the agent modeling process, but the biggest problem found, is that they do not consider the development of virtual organizations.

Therefore, our purpose is to apply the MDD approach to *organizational-oriented* methodologies. In other words, the contribution of this paper is to take as a starting point previous works of organization design in MAS, to develop a *Unified Model of Virtual Organization* which allows flexible implementation (including deployment) on different agent platforms with support for organizations.

3 Modeling Virtual Organizations with MDD

This section presents how to use the MDD approach to model organizations. In order to do this, the core concepts of MDD are presented first. Then, the meta-

models for the organization and the process of creating Virtual Organization using the MDD approach is explained.

3.1 MDD: Core Concepts

The MDD approach uses and creates different models at different abstraction levels, combining them when the application has to be implemented [19]. At high abstraction levels, the models are known as meta-models and they define the structure, semantics and constraints for a family of models (they are the model of a model). Models can be classified into three groups depending on their abstraction level: Computation Independent Model (CIM), which details the general concepts independently whether they are going to be implemented by a machine or not; Platform Independent Model (PIM), which represents the system functionalities without considering the final implementation platform; and the Platform Specific Model (PSM), obtained by combining the PIM with specific details about the selected platform.

A fundamental aspect of the MDD is the definition of sets of transformation rules between models, which allows the models to be automatically converted. Transformations are relational entities that describe how to map the rules concerning how the concepts of one model are transformed into the concepts of another model. These transformations can be applied at different abstraction levels. Horizontal transformations are applied over models that belong to the same level: PIM-to-PIM or PSM-to-PSM. Vertical transformations turn a general model into a more specific one (PIM-to-PSM) [19]. In general, all transformations are known as model-to-model transformations. Additionally, executable code can be automatically generated from a PSM. These transformations are known as model-to-code or model-to-text.

3.2 Using MDD to define an Agent Virtual Organization

One fundamental challenge when defining a platform independent meta-model in an organization is selecting which concepts or components will be included in order to model the organization. It is almost too obvious to mention that this is not a trivial task, since it must define the minimum components necessary for the organization. To achieve this objective, some of the most well-known approaches in the area of MAS organizations were studied (mentioned in Section 2). The purpose of this analysis is to extract the common features from the methodologies studied and adapt them to the current proposal, specifying a platform independent meta-model of the organization. Therefore, the transformation mechanism turns the platform independent model (PIM) into platform specific models (PSM's). Figure 1 shows diagram that illustrates the relationship among the meta-models.

In these methodologies, an agent organization is considered a social entity consisting of a specific number of members which carry out different tasks or functions, and that are structured according to communication patterns and

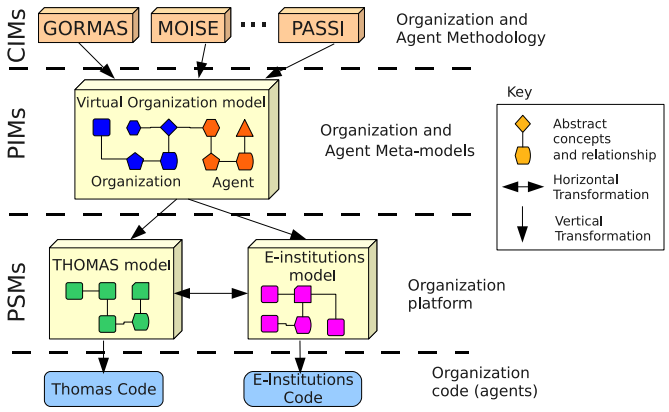


Fig. 1. Using MDD approach in Virtual Organizations

topology specific interactions, to achieve the global objective of the organization, based on behavior rules. The main aspects or factors of an organizations are the structure, functionality, dynamic, normative and its environment. So, to model the characteristics of these components five key concepts are used: *Organizational Unit*, *Service*, *Environment*, *Norm* and *Agent*[3]. These concepts make it possible to represent: (i) how the entities are grouped with each other, defining the relationship between the elements and their environment; (ii) what functionality they offer, including services for the dynamic entry and exit of agents in the organization, and (iii) what restrictions exist regarding the behaviors of system entities[10].

3.3 Organization Meta-model

This section presents our proposal for modeling agent organizations, based on five key concepts(previously mentioned): *Organizational Unit*, *Service*, *Environment*, *Norm* and *Agent*. The proposal is presented defining a set of meta-models, which provide the necessary concepts and primitives to describe structure, functionality, dynamism, environment and normative behaviors of the organization. This set of meta-models will be called π**VOM** (*Platform Independent Virtual Organization Model*) and is structured in different views or perspectives. The motivation for this is to support the evolution of the meta-model and to allow its possible growth in the future. The different views give complementary approaches which, when superposing themselves, generate the complete view of the system. Below, the different views proposed are briefly detailed due to the space limitations of the paper.

Structural view: This view includes all those elements that persist in the organization (see Figure 2(a)). The main concept employed in this view is the Organizational unit (OU). An OU is a group of agents who carry out specific

and differentiated tasks and follow a certain predefined communication and cooperation pattern[4]. This clustering can be seen externally as a single entity that pursues certain objectives, offers and/or requires certain services, and even plays a specific role in order to interact with other entities. Therefore, the organizational unit has a recursive nature and will not only contain agents but also other organizational units acting as atomic entities.

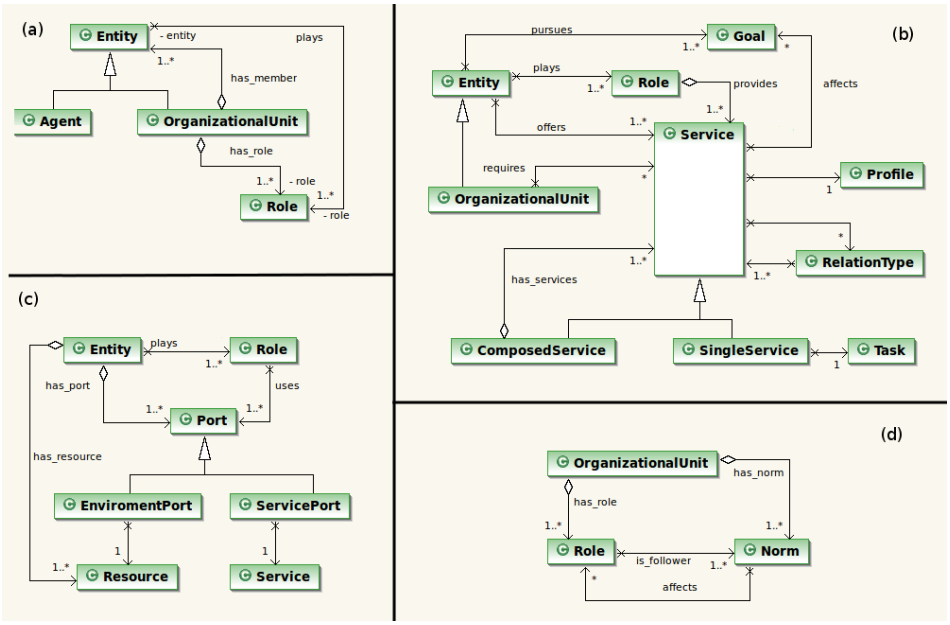


Fig. 2. Concepts used in Strutural(a), Functional(b), Environment(c) and Normative(d) viewpoints of πVOM

Functional view: This view details system functionality, based on services, tasks and objectives. This view shows the general behavior desired by the system. Thus, *Services* represent a certain functionality that an *entity* (Agent or OU) provides other entities, see the meta-model in Figure 2(b). The provider of a *Service* is always associated with a specific *Role*. Service functionality is carried out through the execution of certain *Tasks*, typically executed by the entity providing the service or delegated to other entities. The services can also be composed of several sub-services, and it is possible to define a "workflow" between them using (*RelationType*).

Environment view: This perspective details the elements (*Resources*) of the system. The *resources* are accessed and perceived through an *Environment-Port* (see the meta-model in Figure 2(c)). A *Resource* is an object in the environment that will be consumed by its members. A *Port* represents a point of

interaction between the entity and other elements of the model and serves as an interface to the real world.

Normative view: This model assumes that the organization is managed according to norms. *Norms* are used as mechanisms to limit the autonomy of agents in large systems and solve complex problems of coordination. This view specifies the set of rules and actions defined to control the behavior of members of the organization, specifically the roles of the organization (see Figure 2(d)). These rules correspond to obligations, permissions and prohibitions; also as sanctions and rewards to carry out on its members.

Agent view: A *Agent* is the basic entity of MAS which is within the organization and uses a series of interrelated components, shown in Figure 3. Our *Agent* has a set of basic components: *Capabilities* represent the *know-how* of the *Agent* and follow a pattern of *event-condition-action*. The *Behaviours* implement the roles that the agent can play. The *Task* is the component where the code base of the agent actions is written. For a more detailed explanation of the components of the agent meta-model, please refer to Aguero at el[1].

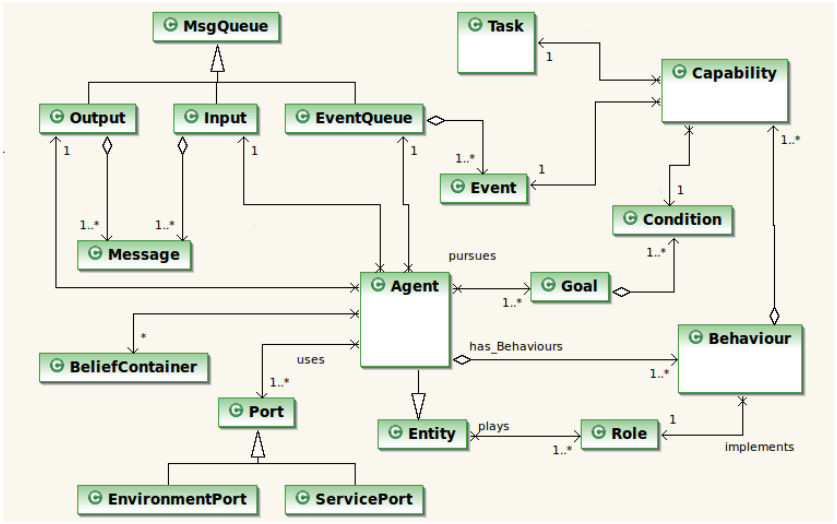


Fig. 3. Concepts used in Agent viewpoint of πVOM

4 Developing an organization with a MDD approach

Once the set of models that characterize our proposed model of *Virtual Organization platform-independent* has been presented, the process for transforming the Virtual Organization into different platforms must be defined. The design process begins by selecting how abstract concepts (which are part of the unified

organization model) are mapped to the target platforms. For this paper, we focus on the study of transformations on two platforms that support agents organizations: THOMAS[7] and E-Institutions[12]. The transformation defines a set of mapping rules. The first mapping rules define which concepts of the source meta-model (πVOM) are transformed to which concepts of the target meta-model, a model-to-model transformation (PIM-to-PSM). This is illustrated by dotted lines in Figure 4. The second transformation translate the models into the code templates of the organization, which can be optionally combined with code written manually by the user. This is a model-to-text transformation (PSM-to-code).

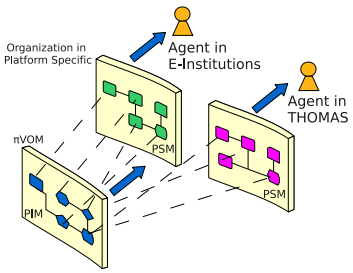


Fig. 4. Tranformation from πVOM to differents platforms

4.1 Development Process Using the THOMAS Platform

The πVOM meta-model is very similar to the model of the organization programmed in THOMAS, since both works are based (partially) on the methodology and artifacts proposed by GORMAS. For this reason, the automatic transformations are relatively easy to describe. Almost all of the abstract concepts of πVOM are represented in THOMAS, so the model-to-model transformation rules are expressed almost as a one-to-one relationship. It is convenient to notice that some concepts in THOMAS have a more detailed feature than πVOM, because THOMAS is a platform-specific model. The transformation rules that must perform the translation between different models are shown in Table 1 (from **rule 1** to **rule 8**).

4.2 Development Process Using the E-Institutions Platform

E-institutions provide a set of tools that are widely used in the area of agents to model organizations, which are defined as a Multi-Agent Systems and can be effectively designed and implemented as *electronic institutions* composed of a vast amount of heterogeneous (human and software) agents playing different roles and interacting by means of speech acts[12]. They take inspiration from traditional human institutions, and offer a general agent-mediated computational model

Rule	Concept	Transformation
	THOMAS	
1	Organizational Unit	πVOM.OU $\Rightarrow$ THOMAS.OU
2	Agent	πVOM.Agent $\Rightarrow$ THOMAS.Agent
3	Role	πVOM.Role $\Rightarrow$ THOMAS.Role
4	Service	πVOM.Service $\Rightarrow$ THOMAS.Service
5	Norm	πVOM.Norm $\Rightarrow$ THOMAS.Norm
6	RelationType	πVOM.RelationType $\Rightarrow$ THOMAS.Process
7	Resource	πVOM.Resource $\Rightarrow$ THOMAS.Resource
8	Goal	πVOM.Goal $\Rightarrow$ THOMAS.Goal
	E-INSTITUTIONS	
9	Organizational Unit	$\bigcup \pi$VOM.OU $\Rightarrow \bigcup$ EI.Scene
10	Agent	πVOM.Agent $\Rightarrow$ EI.Agent
11	Role	πVOM.Role $\Rightarrow$ EI.Role
12	Service	$\forall \pi$VOM.Service $\in$ OU $\Rightarrow \bigcup$ EI.State $\in$ Scene
13	RelationType	πVOM.RelationType $\Rightarrow$ EI.Transition OR EI.Illocutions
14	Norm	πVOM.Norm $\Rightarrow$ EI.Norm
15	Goal	πVOM.Goal $\Rightarrow$ EI.Norm
16	Resource	πVOM.Resource $\Rightarrow$ EI.World

Table 1. Rules from π *VOM* to THOMAS and E-Institutions

that serves to realize an *agent-mediated electronic institutions*. In Figure 5, the main meta-model components of E-Institutions are presented. Transformation rules for E-Institutions are shown in Table 1 (from **rule 9** to **rule 16**). Due to the space limitations of this paper, each transformation rule cannot be detailed. Instead, in order to illustrate its use to the reader, an application scenario of core rules is described in the next section.

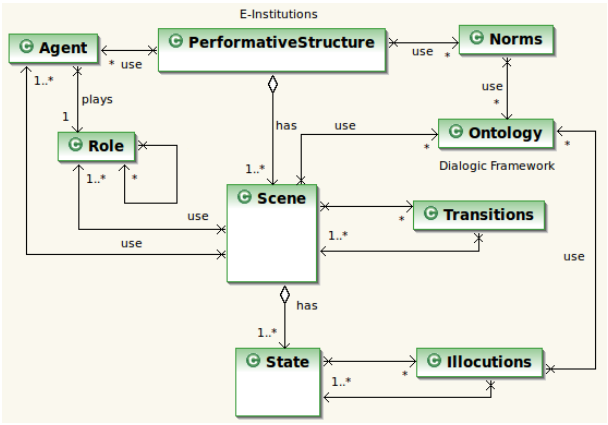


Fig. 5. Core concepts used in E-Institutions

4.3 Usage Scenario

To illustrate the use of the rules, a case scenario for making flight and hotel arrangements is utilized. This is a well known example that has been modeled

by means of electronic institutions in previous works (Dignum [11]; Argente et al [3]). The *Travel Agency* example is an application that facilitates the interconnection between clients (individuals, companies, travel agencies) and providers (hotel chains, airlines); delimiting services that each one can request or offer. The system controls which services must be provided by each entity. Provider entities are responsible for the internal functionality of these services. However, the system imposes some restrictions on service profiles, service requesting orders and service results. In this system, agents can search for and make hotel and flight reservations, and pay in advance for bookings. This case study is modeled as an organization (*TravelAgency*) inside which there are two organizational units (HotelUnit and FlightUnit) which represent a group of agents. Each unit is dedicated to hotels or flights, respectively. Three kind of roles can interact in the Travel Agency example: customer, provider and payee roles. The Customer role requests system services. More specifically, it can request hotel or flight search services, booking services for hotel rooms or flight seats, and payment services. The Provider role is in charge of performing the service. Finally, the Payee role gives the advanced payment service. Figure 6 shows the *TravelAgency* structure, with its units, roles and relationships between them.

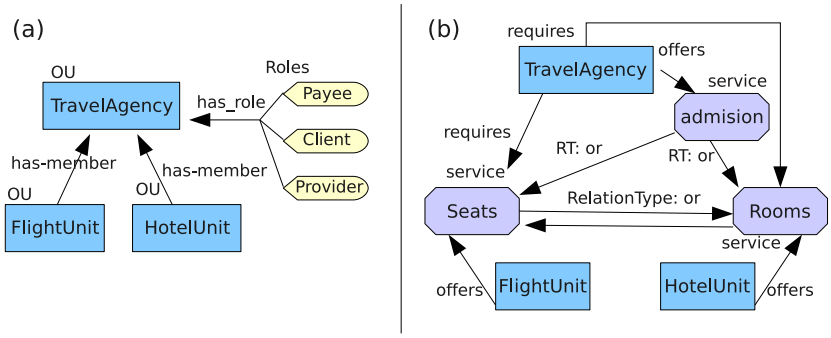


Fig. 6. Travel Agency in πVOM

The process begins by modeling the Travel Agency (structural and functional models, see Figure 6(a)(b)), and applying the rules mentioned previously, in order to obtain the organizations on THOMAS and E-Institutions. In the case of THOMAS the models are very similar and their transformation in almost direct (see in Figure 7(a)). But, getting the organization in E-Institutions and creating the core templates of *PerformativeStructure* seen in figure 7(b), requires the application of the **rule 9** and **rule 13**.

Rule 9 allows all of the *Scenes* in the E-Institutions that correspond to the *OU* of πVOM (3 in total) to be obtained. Additionally the *Scenes* for entrance and exit are needed (root and output). After the application of **Rule 13**, as the *Services* are composed in the order given by the *RelationType*, we can specify the type of *Transitions* between the different *Scenes* in the E-Institutions, which in

this case correspond to *Transitions* of type OR. As previously mentioned, this mapping generates the basic template of the PerformativeStructure of the E-Institutions (Figure 7(b)). With the application of the remaining rules, a more detailed description of the PerformativeStructure is obtained, but due to the limitations of space in this paper, they are not explained.

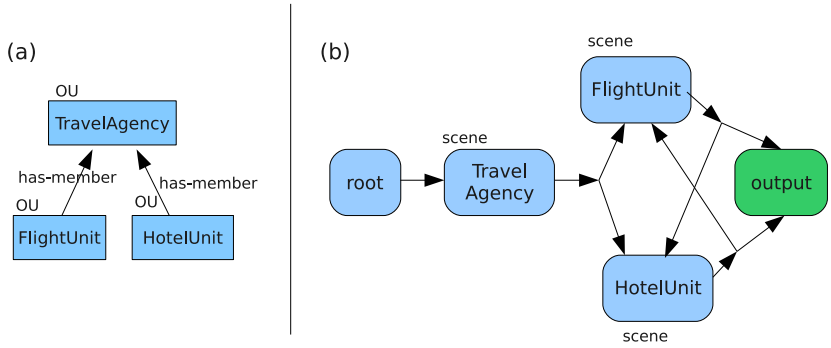


Fig. 7. Travel Agency in THOMAS and E-Institutions

5 Conclusions

This work presents a meta-model of Virtual Organization (called πVOM) which allows organizations in MAS to be modeled using abstract components which are independent of the implementation platform following a MDD approach. This meta-model is divided into five views that focus on the most important aspects of the Virtual Organizations and these views can easily be extended if required to a specific domain.

From unified meta-model, creating code templates for specific platforms of organizations is possible. This was discussed and exemplified using transformations on the THOMAS and E-Institutions platforms. These transformations show that the meta-model can be considered platform independent. As future work, we will propose additional transformations in order to obtain the agent instances (to generate the agent code) in the proposal and other frameworks. We will propose the introduction of specific components/views (additional) to reach agreements. We propose that the Virtual Organization provides the framework and components required to model agreements, or in other words, use the MDD approach to drive design-oriented agreements.

Acknowledgment: This work was partially supported by TIN2006-14630-C03-01 and PROMETEO/2008/051 projects of the Spanish government and CONSOLIDER-INGENIO 2010 under grant CSD2007-00022.

References

1. J. Agüero, M. Rebollo, C. Carrascosa, and V. Julián. Does android dream with intelligent agents? In *Int. Symposium on Distributed Computing and Artificial Intelligence 2008(DCAI2008)*, ISBN: 978-3-540-85862-1, pages 194–204, 2008.
2. J. Agüero, M. Rebollo, C. Carrascosa, and V. Julián. Agent design using Model Driven Development In *7th Int. Conference on Practical Applications of Agents and Multi-Agent Systems(PAAMS2009)*, ISBN 978-3-642-00486-5, pages 60–69, 2009.
3. E. Argente, V. Julian, and V. Botti. Mas modelling based on organizations. In *9th Int. Workshop on Agent Oriented Software Engineering*, pages 1–12, 2008.
4. E. Argente Villaplana. *Gormas: Guías para el desarrollo de sistemas multiagente abiertos basados en organizaciones*. PhD thesis, UPV, Spain, 2008.
5. B. Bauer. Uml class diagrams revisited in the context of agent-based systems. *Proceedings Agent-Oriented Software Engineering*, pages 101–118, 2002.
6. P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.
7. C. Carrascosa, A. Giret, V. Julian, M. Rebollo, E. Argente, and V. Botti. Service oriented multi-agent systems: An open architecture. In *Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1–2, 2009.
8. R. Cervenka and I. Trencansky. *The Agent Modeling Language – AML*. Whitestein Series in Software Agent Technologies and Autonomic Computing, 2007.
9. M. Cossentino and C. Potts. Passi: A process for specifying and implementing multi-agent systems using uml. Technical report, University of Palermo, 2001.
10. N. Craido, E. Argente, V. Julián, and V. Botti. Designing virtual organizations. In *7th Int. Conference on Practical Applications of Agents and Multi-Agent Systems(PAAMS2009)*, ISBN 978-3-642-00486-5, pages 440–449, 2009.
11. V. Dignum. *A model for organizational interaction: based on agents, founded in logic*. Phd dissertation, Utrecht University, 2003.
12. M. Esteva, J. A. Rodríguez-Aguilar, C. Sierra, and J. L. Arcos. On the formal specifications of electronic institutions. *Lecture Notes in Computer Science*, pages 126–147, 2001.
13. J. Ferber, O. Gutknecht, and F. Michel. From agents to organizations: an organizational view of multi-agent systems. *Lecture Notes in Computer Science*, pages 214–230, 2004.
14. I. Garca-Magario, J. Gómez-Sanz, and R. Fuentes. Ingenias development assisted with model transformation by-example: A practical case. In *7th Int. Conf. on Practical Applications of Agents and MultiAgent Systems*, pages 40–49, 2009.
15. B. Gateau, O. Boissier, D. Khadraoui, and E. Dubois. Moiseinst: An organizational model for specifying rights and duties of autonomous agents. In *Third European Workshop on Multi-Agent Systems (EUMAS2005)*, pages 484–485, 2005.
16. Z. Guessoum and T. Jarraya. Meta-models & model-driven architectures. In *Contribution to the AOSE TFG AgentLink3 meeting*, 2005.
17. C. Hahn, C. Madrigal-Mora, and K. Fischer. A platform-independent meta-model for multiagent systems. In *Autonomous Agents and Multi-Agent Systems*, 18(3):239–266, 2009.
18. N. Jennings and M. Wooldridge. Applications of intelligent agents. In *Agent Technology: Foundations, Applications, and Markets*, pages 3–28, 1998.
19. A. Kleppe, J. B. Warmer, and W. Bast. *MDA Explained: The Model Driven Architecture: Practice and Promise*. Addison-Wesley Professional, 2003.

A Logic Related to Minimal Knowledge ⋆

David Pearce[1] and Levan Uridia[2]

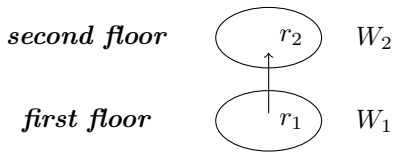
[1] Universidad Politcnica de Madrid, Spain, david.pearce@upm.es
[2] Universidad Rey Juan Carlos, uridia@ia.urjc.es

Abstract. We introduce and study a modal logic $wK4F$ which is closely related to the logic $S4F$ that is important in the context of epistemic logics for representing and reasoning about an agent's knowledge. It is obtained by adding the axiom F to the modal logic $wK4$, or dropping from $S4F$ the T axiom. We show that $wK4F$ is sound and complete with respect to the class of all minimal topological spaces i.e. topological spaces with only three open sets. We characterize the rooted frames of the logic $wK4F$ by quadruples of natural numbers and in the same fashion we give a characterization of p-morphic images of rooted $wK4F$ frames.

1 Introduction

In epistemic logic and formalisms for representing an agent's knowledge, the paradigm of *minimal knowledge* has played an important role [1]. $S4F$ is a reflexive normal modal logic first introduced and studied by Segerberg [2]. Later its importance for knowledge representation and reasoning was discovered by Truszczynski [3] and Schwarz and Truszczynski [4] who used it to investigate the idea of minimal knowledge. They showed that the nonmonotonic version of the logic $S4F$ captures, under some intuitive encodings, several important approaches to knowledge representation and reasoning. They include disjunctive logic programming under answer set semantics [5], (disjunctive) default logic [6]; [7], the logic of grounded knowledge [8], the logic of minimal belief and negation as failure [10] and the logic of minimal knowledge and belief [4]. Recently, Truszczynski [11] and Cabalar [12] have revived the study of $S4F$ in the context of a general approach to default reasoning.

In terms of Kripke models, $S4F$ is captured by frames consisting of two clusters of points connecting by an accessibility relation:



⋆ Partially supported by the MCICINN projects TIN2006-15455-CO3 and CSD2007-00022.

The points in each cluster, W_1, W_2, are reflexive. In this paper we study a logic closely related to $S4F$, called $wK4F$. It is obtained by dropping from $S4F$ the T axiom and therefore the condition of reflexivity on points; so, while the basic picture is the same, some points in W_1, W_2 are now irreflexive.

Since $S4F$ and $wK4F$ are closely related, many results about the latter can be transferred to the former. In this paper we examine $wK4F$, prove a completeness theorem and show that there is a close relation between $wK4F$ and the class of *minimal* topological spaces. We leave for future work the study of the non-monotonic variant of $wK4F$ as well as multi-modal versions that may be used to represent common knowledge. However we point out that by a standard translation technique there is a natural embedding of $wK4F$ into $S4F$. The embedding is obtained by so called splitting translation the main clause of which is given by $Sp(\Box\phi) = \Box\phi \wedge \phi$.

The paper is organized in the following way. In section 2 we present the syntax and kripke semantics of the modal logic $wK4F$. We prove the completeness and finite model property with respect to given semantics. In section 3 we give the characterization of finite one-step, weak-transitive frames and their bounded morphisms in terms of quadruples of natural numbers. In section 4 we prove the main theorem of the paper, which states that $wK4F$ is the complete and sound logic of all minimal topological spaces. The last section gives the conclusion and questions for future work.

2 The modal logic $wK4F$

Following Tarski's suggestion to treat modality as the derivative of the topological space, Esakia [13] introduced the modal logic $wK4$ as the modal logic of all topological spaces, with the desired (derivative operator) interpretation of the modal $\Diamond$. The modal logic $wK4F$ is a normal modal logic obtained by adding the axiom weak-F to the modal logic $wK4$. $wK4F$ is a weaker logic then $S4F$ discussed in Segerberg [2]. In particular we get the modal logic $S4F$ by adding axiom T to the logic $wK4F$. Thus in this sense $wK4F$ is of interest for $S4F$ also as some results are easily carried over and simplified from one to another.

2.1 syntax

Definition 2.11 *The normal modal logic $wK4F$ is defined in a standard modal language with an infinite set of propositional letters $p, q, r..$ and connectives $\vee, \wedge, \Box, \neg$,*

- *The axioms are all classical tautologies plus the following axioms:*

$\Box\top \leftrightarrow \top$,
$\Box(p \wedge q) \leftrightarrow \Box p \wedge \Box q$,
$\Box p \wedge p \rightarrow \Box\Box p$,
$p \wedge \Diamond(q \wedge \Box\neg p) \rightarrow \Box(q \vee \Diamond q)$.

- *The rules of inference are: Modus ponens, Substitution and Necessitation.*

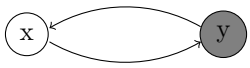
2.2 Kripke semantics

The Kripke semantics for the modal logic $wK4F$ is provided by one-step, weak-transitive Kripke frames. Below we give the definition of these frames.

Definition 2.21 *We will say that a relation $R \subseteq W \times W$ is weak-transitive if* $(\forall x, y, z)(xRy \wedge yRz \wedge x \neq z \Rightarrow xRz)$.

Of course every transitive relation is weakly-transitive also. Moreover it is not difficult to see that weakly-transitive relations differ from transitive ones just by the occurrence of irreflexive points in clusters. As one can see, the frame in the picture is weakly transitive, but not transitive.

The picture represents the diagramatic view of kripke structure. Irreflexive points are colored by grey and reflexive points are uncolored. Arrows represent the relation between two distinct points. So as we can see yRx and xRy, but $y \not Ry$, which contradicts transitivity, but not weak transitivity as $y = y$.



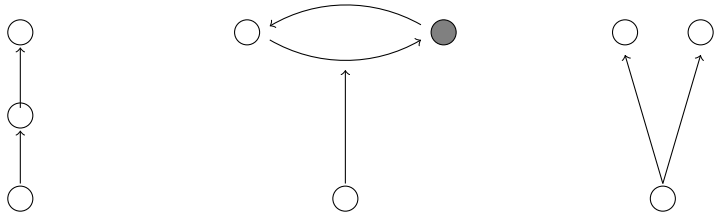
pic. 1

Definition 2.22 *We will say that a relation $R \subseteq W \times W$ is a one-step relation if the following two conditions are satisfied:*

$1)(\forall x, y, z)((xRy \wedge yRz) \Rightarrow (yRx \vee zRy))$,

$2)(\forall x, y, z)((xRy \wedge \neg(yRx) \wedge xRz) \Rightarrow (zRx \vee (zRy \vee yRz)))$.

As the reader can see the first condition restricts the "strict" height of the frame to two. Where informally by "strict" we mean that the steps are not counted within a cluster. So for example we cannot have the situation on the left hand side of the picture below, while we can have the situations in the middle and on the right hand side.



The second condition is more complicated. Nevertheless it is not too hard to verify that it restricts the "strict" width of the frame to one. So again we cannot have for example the figure on the right hand side, while the figures in the middle and on the left are allowed. Here we cheat slightly as indeed the frame is not allowed to fork as in the picture on the right, but condition 2) does not restrict the reversed fork i.e. frame with two points on the bottom and one on the top. So strict width does not effect points on the bottom.

For the sake of completeness we will just briefly recall the main definitions in modal logic, like: Kripke frame, satisfaction and validity of modal formulas.

Definition 2.23 *The pair (W, R), with W an arbitrary set (set of possible worlds) and $R \subseteq W \times W$ is called a Kripke frame.*

If we additionally have a third component $V : Prop \times W \to \{0, 1\}$, then we say that we have a Kripke model $M = (W, R, V)$ (where Prop denotes the set of all propositional letters).

The satisfaction and validity of a modal formula are defined inductively. We just state the base and modal cases here.

Definition 2.24 *For a given kripke model $M = (W, R, V)$ the satisfaction of a formula at a point $w \in W$ is defined inductively as follows: $w \Vdash p$ iff $V(p, w) = 1$, the boolean cases are standard, $w \Vdash \Box\phi$ iff $(\forall v)(wRv \Rightarrow v \Vdash \phi)$.*

Note: From the definition of $\Diamond$, $(\Diamond\phi \equiv \neg\Box\neg\phi)$ it follows that $w \Vdash \Diamond\phi$ iff $(\exists v)(wRv \wedge v \Vdash \phi)$.

So far we defined the modal logic $wK4F$ syntactically and we gave the definition of one-step and weakly-transitive Kripke frames. The following two propositions link these two things. The proof uses standard modal logic completeness techniques, so we will not enter into all the details.

Proposition 2.25 *The modal logic $wK4F$ is sound w.r.t. the class of all one-step and weakly-transitive Kripke frames.*

Proof. We give the proof only for the fourth axiom of the modal logic $wK4F$. The proofs for other axioms are analogous. The reader may also consult [13].

Take an arbitrary, weakly-transitive, one-step frame (W, R). Take an arbitrary valuation V and assume at some point $w \in W$ that $w \Vdash p \wedge \Diamond(q \wedge \Box\neg p)$. By the definition of satisfaction this implies that $w \Vdash p$ and there exists w' such that wRw', $w' \Vdash q$ and it is not the case that $w'Rw$ as much as $w' \Vdash \Box\neg p$. Now take an arbitrary v such that wRv. By the second condition in the definition of one-step relation we have that either vRw or $vRw' \wedge w'Rv$. If $v = w'$ we immediately have that $v \Vdash q$ since $w' \Vdash q$. In case $v \neq w'$ we have two subcases $vRw' \wedge w'Rv$ or $vRw \wedge wRv$. In both cases by weak-transitivity of the relation R we have at least vRw', which implies that $v \Vdash \Diamond q$ and hence $w \Vdash \Box(q \vee \Diamond q)$.

Proposition 2.26 *The modal logic $wK4F$ is complete w.r.t. the class of all **finite**, one-step and weakly-transitive Kripke frames.*

Proof. We do not give the details here as far as the proof uses standard techniques given in [13]. First we take the canonical model for $wK4F$ and then apply the minimal filtration to it.

3 The class of finite, rooted, weakly-transitive and one-step Kripke frames and their bounded morphisms.

We have seen that the class of finite, weakly-transitive and one-step Kripke frames fully captures the modal logic $wK4F$. This class can be reduced to a smaller class of frames which are rooted. In this section we characterise finite, rooted, weakly-transitive and one-step Kripke frames and their bounded morphisms in terms of quadruples of natural numbers.

Definition 3.07 *The upper cone of a set $A \subseteq W$ in a Kripke frame (W, R) is defined as a set $R(A) = \bigcup\{y : x \in A \& xRy\}$.*

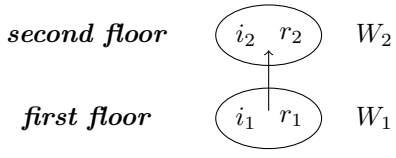
Definition 3.08 *A Kripke frame (W, R) is called* rooted *if there exists a point $w \in W$ such that the upper cone $R(\{w\}) = W$; w is called the root of the frame.*

Note 1. The root is not necessarily unique. For example on the first picture both x and y are roots.

Let N^4 be the set of all quadruples of natural numbers and let $\mathcal{N}^4 = N^4 - \{(n, m, 0, 0) | n, m \in N\}$. The following theorem states that the set of finite, rooted, one-step, weakly-transitive frames can be viewed as the set $\mathcal{N}^4$. Let K denote the class of all rooted, finite, one-step, weakly-transitive frames considered up to isomorphism.

Theorem 3.09 *There is a one-to-one correspondence between the set* K *and the set $\mathcal{N}^4$.*

Proof. We know that any one-step frame has "strict" width one and "strict" height less than or equal to two (We didn't give the formal definition of "strict" height and width, but it should be clear from the intuitive explanation after the definitions 2.22 and 2.21 what we mean by this). If additionally we have that the frame is rooted, the case where width is greater than one at the bottom is also restricted. It is not difficult to verify that any such frame (W, R) is of the form (W_1, W_2), where $W_1 \cup W_2 = W$, $W_1 \cap W_2 = \varnothing$ and $(\forall u \in W_1, \forall v \in W_2)(uRv)$. Besides because of the weak-transitivity, we have that $(\forall u, u' \in W_1)(u \neq u' \Rightarrow uRu')$ and the same for every two points $v, v' \in W_2$. We will call W_1 the ***first floor*** and W_2 the ***second floor*** of the frame (W, R). Pictorially any rooted, weak-transitive and one-step Kripke frame can be represented as in the picture below.



Note 2. It is possible that W_1 or W_2 is equal to $\varnothing$ i.e. the frame has only one floor. In this case we treat the only floor of the frame as the second floor.

Now let us describe how to construct the function from K to $\mathcal{N}^4$. With every frame $(W, R) \in \mathcal{N}^4$ we associate the quadruple (i_1, r_1, i_2, r_2), where i_1 is the number of irreflexive points in W_1, r_1 is the number of reflexive points in W_1, i_2 is the number of irreflexive points in W_2 and r_2 is the number of reflexive points in W_2. We will call the quadruple (i_1, r_1, i_2, r_2) the ***characteriser*** of the frame (W, R). In case the frame (W, R) has only one floor, by note 2 above it is treated as the frame $(\varnothing, W)$. Hence it's characteriser has the form $(0, 0, i, r)$. Now it is clear that the correspondence described above defines a function from the set K to the set $\mathcal{N}^4$. Let us denote this function by Ch.

claim 1: Ch is injective. Take any two distinct finite, rooted, weakly-transitive, one-step Kripke frames (W, R) and (W', R'). That they are distinct in K means that they are non-isomorphic i.e. either $|W| \neq |W'|$ or $R \not\simeq R'$. In the first case it is immediate that $Ch(W, R) \neq Ch(W', R')$ since $|W| = i_1 + i_2 + r_1 + r_2$. In the second case we have three subcases:

1) $|W_1| \neq |W_1'|$. In this subcase $i_1 + r_1 \neq i_1' + r_1'$ and hence $Ch(W, R) \neq Ch(W', R')$.

2) The number of reflexive (irreflexive) points in $|W_1|$ differs from the number of reflexive (irreflexive) points in $|W_1'|$. In this subcase $i_1 \neq i_1'$ and again $Ch(W, R) \neq Ch(W', R')$.

3) The number of reflexive (irreflexive) points in $|W_2|$ differs from the number of reflexive (irreflexive) points in $|W_2'|$. This case is analogous to the previous.

It is straightforward to see that if none of these cases above occur i.e. $|W| = |W'|$, $|W_1| = |W_1'|$, $|\{w | w \in W_1 \wedge wRw\}| = |\{w' | w' \in W_1' \wedge w'R'w'\}|$ and $|\{w | w \in W_2 \wedge wRw\}| = |\{w' | w' \in W_2 \wedge w'R'w'\}|$ then (W, R) is isomorphic to $(W'R')$ and hence $(W, R) = (W', R')$ in K.

claim 2: Ch is surjective. Take any quadruple $(i_1, r_1, i_2, r_2) \in \mathcal{N}^4$. Let us show that the pre-image $Ch^{-1}((i_1, r_1, i_2, r_2))$ is not empty. Take the frame $(W, R) = (W_1, W_2)$, where $|W_1| = i_1 + r_1$, $|W_2| = i_2 + r_2$, W_1 contains i_1 irreflexive and r_1 reflexive points and $|W_2|$ contains i_2 irreflexive and r_2 reflexive points. Then by the definition of Ch, we have that $Ch(W, R) = (i_1, r_1, i_2, r_2)$.

Definition 3.010 *The function f between two frames (W, R) and (W', R') is called a bounded morphism if the following two conditions are satisfied:*
(1) $wRv \Rightarrow f(w)R'f(v)$,
(2) $f(w)R'v' \Rightarrow (\exists v \in W)(wRv \wedge f(v) = v')$.

The bijection given in theorem 3.09 can be extended to bounded morphisms. In the following theorems we characterise the bounded morphisms between two finite, rooted, weakly-transitive, one-step frames in terms of conditions on the quadruples of natural numbers. We split the proof into three different theorems to make it much more readable; besides each case is interesting on its own. In the first theorem we will give the characterisation for frames with the characterisers $(0, 0, i, r)$ i.e. for frames with only one floor, then we will give the characterisations for frames where one has two floors and the second is one floor frame and the third theorem will give the condition for frames where both have two floors. We will omit zeroes in the quadruple $(0, 0, i, r)$ and just write it as (i, r).

Theorem 3.011 *The finite, rooted, weakly-transitive, one-step frame* (W', R') *with the characteriser* (i', r') *is a bounded morphic image of the finite, rooted, weakly-transitive, one-step frame* (W, R) *with the characteriser* (i, r) *iff the following conditions are satisfied:*

$r' = 0 \Rightarrow (i, r) = (i', r')$,
$i \geq i'$,
$2 \times (r' - r) \leq i - i'$.

Note that the operation minus is defined within the natural numbers i.e. $n - m = 0$ *if* $m > n$.

Proof. For the left to right direction assume $f : (W, R) \twoheadrightarrow (W', R')$. This means that $i + r \geq i' + r'$, since$f$ is a surjection. First let us state some general observations which will help in proving the theorem.

• ***for every irreflexive point*** $w' \in W'$***, we have that*** $f^{-1}(w')$ ***is one irreflexive point.*** Assume not. Then either $f^{-1}(w')$ contains some reflexive point $w \in W$, or it contains at least two irreflexive points $u, v \in W$. In the first case we have wRw but not $f(w)R'f(w)$, so we obtain a contradiction. In the second case we have $uRv \wedge vRu$ but not $f(v)R'f(u)$ and again this contradicts f being a bounded morphism. Now we are ready to begin the proof of the theorem. Let us check that all conditions are satisfied.

case 1 Assume $r' = 0$ but $(i, r) \neq (i', r')$. So either $r \neq 0$ or $i \neq i'$. In both cases we get a contradiction by the above observation, as reflexive points cannot be mapped to irreflexive ones and also two irreflexive points can not be mapped to one irreflexive point.

case 2 Assume $i < i'$. Then there is at least one point $v' \in W'$ such that $f^{-1}(v') = \varnothing$. The reason is that there are not enough irreflexive points in W to cover all irreflexive points in W'. And we know (by above remarks) we cannot map reflexive points to irreflexive ones. So we get a contradiction.

case 3 Assume $2 \times (r' - r) > i - i'$. This means that $r' > r$. So there are $r' - r$ reflexive points in W' with the pre-image not containing a reflexive point. But then there is at least one reflexive point $w' \in W'$ such that $f^{-1}(w')$ contains less than 2 irreflexive points. This is because by assumption there are not enough pairs of irreflexive points in W for all reflexive points with pre-image not containing reflexive ones. But this gives a contradiction because either f is not surjective (in case $f^{-1}(w') = \varnothing$) or f is not a bounded morphism (in case $f^{-1}(w') = v$ with v irreflexive).

Now let us prove the converse direction. Let us enumerate points in W in the following way: Let $w_1, ...w_r$ be the reflexive points and $v_1, ...v_i$ irreflexive points. Let us use the same enumeration for points in W' with the difference that we add $'$ to every point. So for example w'_1 is the reflexive and v'_2 is the irreflexive point in W'. In case $r' = 0$ we know that $(i, r) = (i', r')$ and we can take f to be bijection mapping w_i to w'_i.

In case $r' \neq 0$ we distinguish two subcases.

case 1 When $r > r'$. Let us define $f : W \to W'$ in the following way:

$f(v_j) = v'_j$ for $j \in \{1, .., i'_1\}$,
$f(w_j) = w'_j$ for $j \in \{1, .., r'_1 - 1\}$,
$f(v_{i'+1}) = f(v_{i'+2}) = ... = f(v_i) = f(w_{r'+1}) = .. = f(w_r) = w'_r$.

case 2 When $r \leq r'$. Let us define $f : W \to W'$ in the following way:

$f(v_j) = v'_j$ for $j \in \{1, .., i'\}$,
$f(w_j) = w'_j$ for $j \in \{1, .., r\}$,
$f(v_{i'+2k-1}) = f(v_{i'+2k}) = w'_{r+k}$ for $k \in \{1, .., r' - r - 1\}$,
$f(v_{i'+2(r'-r)-1}) = f(v_{i'+2(r'-r)}) = .. = f(v_i) = w'_r$.

In other words we send each reflexive point $w_j \in W$ to the reflexive point $w'_j \in W'$ and each irreflexive point $v_j \in W$ to the irreflexive point $v'_j \in W'$. Inasmuch as we have that $i \geq i'$ and $r \leq r'$, there may be left some irreflexive points in W on which we have not yet defined f and also some reflexive points in W' which have no pre-image, so we associate to every pair of such irreflexive points one reflexive point which has no pre-image. We continue this process until we are left with only one reflexive point without pre-image (we know this exists by the condition $r' \neq 0$) and associate to it all the remaining irreflexive points on which f was not defined. The condition $2 \times (r - r') \leq i' - i$ guarantees that there are at least two such irreflexive points left. It is easy to check that f defined in the following way is indeed a bounded morphism.

Now let us consider the case, where the first frame has two floors, while the second is a one floor frame. Note that the only difference from the conditions in 3.011 is that we require that the second frame contains at least one reflexive point. This is because we want to map all first floor points of the first frame to this particular reflexive point.

Theorem 3.012 *The finite, rooted, weakly-transitive, one-step frame* (W', R') *with the characteriser* (i', r') *is a bounded morphic image of the finite, rooted, weak-transitive, one-step frame* (W, R) *with the characteriser* (i_1, r_1, i_2, r_2)*, where* $i_1 + r_1 > 0$ *and* $i_2 + r_2 > 0$ *iff the following conditions are satisfied:*

$r' \neq 0$,
$i_2 \geq i'$,
$2 \times (r' - r_2) \leq i_2 - i'$.

Proof. Assume $r' = 0$. This means that all points in (W', R') are irreflexive. Now as (W, R) has two floors we can represent it as a pair (W_1, W_2) where both sets are non-empty. This implies that there are at least two points $u \in W_1$ and $v \in W_2$, such that $f(u) \in W'$ and $f(v) \in W'$. Since (W', R') is a cluster, we have that $f(u)R'f(v)$ and $f(v)R'f(u)$. But this contradicts the property that f is a bounded morphism. This is because not Ru while $f(v)R'f(u)$ and as $f(u)$

is irreflexive there does not exist a point $w \in W$, with $u \neq w$ and $f(w) = f(v)$ (see the first observation in the proof of lemma 3.011).

Assume $i_2 < i'$. Again by first observation in the proof of lemma 3.011 we know that the pre-image of the irreflexive point cannot be reflexive. This means that there exists a point $u' \in W'$, such that, u' is irreflexive and $f^{-1}(u') \cap W_2 = \varnothing$. Different from the analogous case in the proof of lemma 3.011 the possibility arises that some irreflexive point $u \in W_1$ is mapped to the point u'. Let us show that this cannot happen. Assume $u \in W_1$ and $f(u) = u'$. As u is a first floor point, there exists some $v \in W_2$ with uRv. As f is bounded morphism we have that $f(u)R'f(v)$. As (W', R') is a cluster we have that $f(v)R'f(u)$ as well. Now we get a contradiction as there is no successor of v which is mapped to $f(u) = u'$.

Assume the third condition does not hold. This means that $2 \times (r' - r_2) > i_2 - i'$. In this case we have that there is at least one point $u' \in W'$ such that u' is reflexive and $f^{-1}(u') \cap W_2$ is either the empty set or it contains only one point u with $u \not R u$. This gives a contradiction, since $u'R'u'$ while there is no successor v of u with $f(u) = u'$.

For the converse direction we construct $f : W \twoheadrightarrow W$ in the following way: $f_{|W_1} = v'$, where v' is an arbitrary reflexive point in W' (we know that such a point exists from the first condition of the lema). $f_{|W_2}$ is constructed in exact analogy with the construction in 3.011.

And at last we can give the characterisation for the case where both frames have two floors.

Theorem 3.013 *The finite, rooted, weakly-transitive, one step-frame (W', R') with the characteriser (i'_1, r'_1, i'_2, r'_2), where $i'_1 + r'_1 > 0$ and $i'_2 + r'_2 > 0$ is a bounded morphic image of the finite, rooted, weakly-transitive, one step-frame (W, R) with the characterisers (i_1, r_1, i_2, r_2), where $i_1 + r_1 > 0$ and $i_2 + r_2 > 0$ iff the following hold:*

$r'_1 = 0 \Rightarrow (i_1, r_1) = (i'_1, r'_1)$,
$r'_2 = 0 \Rightarrow (i_2, r_2) = (i'_2, r'_2)$,
$i_1 \geq i'_1$,
$i_2 \geq i'_2$,
$2 \times (r'_1 - r_1) \leq i_1 - i'_1$,
$2 \times (r'_2 - r_2) \leq i_2 - i'_2$.

The operation minus is defined within the natural numbers i.e. $n - m = 0$ if $m > n$.

Proof. The theorem follows from the previous theorem and the following observation:

• ***If*** (W', R') ***is a two floor frame i.e.*** $i'_1 + r'_1 > 0$ ***and*** $i'_2 + r'_2 > 0$ ***then*** (W, R) ***is also two floor frame.*** Assume not. Then there exist points $w, v \in W$ such that vRw and $f(v) \not R' f(w)$ as $f(v)$ is a second floor point while $f(w)$ is a first floor point.

• ***points from the second floor cannot be mapped to points on the first floor.*** For the contradiction assume that the point $f(w)$ is a first floor point while w is a second floor point. This means that there exists $v' \in W'$ such that $f(w)R'v'$ and not $v'R'f(w)$. Then as f is a bounded morphism there exists $v \in W$ such that wRv and $f(v) = v'$. As w is a second floor point, $wRv \Leftrightarrow vRw$ and we get a contradiction as we have vRw while not $f(v)R'f(w)$.

• ***points from the first floor cannot be mapped to the points on the second floor.*** For the contradiction assume that the point $f(w)$ is a second floor point while w is a first floor point. Now either there is another point $w_1 \in W$ on the first floor with $f(w_1)$ also on the first floor or all points including w from the first floor of W are mapped to the second floor of W'. In the first case wRw_1 and not $f(w)Rf(w_1)$ so we get a contradiction. In the second case we get that f is not surjective inasmuch as both frames were supposed to be two floor frames so there is at least one point on the first floor in W' left with empty pre-image. This is because by above observation second floor points cannot be mapped to first floor points.

The converse direction is proved just by repetition of the case for one floor frames for the other floor.

4 Connection with minimal topological spaces

In this section we show that that the modal logic $wK4F$ is the modal logic of minimal topological spaces. A topological space is minimal if it has only three open sets. It is well known that there is a bijection between Alexandrof spaces and weakly transitive, irreflexive Kripke frames. It is also well known that this bijection preserves modal formulas. In this section we show that the special case of this correspondence for minimal topological spaces gives one step, irreflexive and weakly transitive relations as a counterpart. As a corollary it follows that the logic $wK4F$ is sound and complete w.r.t. the class of minimal topological spaces.

Theorem 4.014 *There is a one-to-one correspondence between the class of* ***irreflexive****, weakly-transitive, finite, rooted, one-step Kripke frames and the class of all finite minimal topological spaces.*

Proof. Assume (W, R) is a finite, rooted, weakly transitive and one step relational structure and besides R is irreflexive. (Note that as the frame is irreflexive its characteriser has the form $(i_1, 0, i_2, 0)$, where $i_1 + i_2 = |W|$.) Let W_1 be the first floor and W_2 the second floor of the frame, then the topology we construct is $\{W, \varnothing, W_2\}$. It is immediate that the space (W, Ω_R), where $\Omega_R = \{W, \varnothing, W_2\}$, is a minimal topological space.

Let us show that the correspondence we described is injective. Take two arbitrary distinct irreflexive, finite, rooted, weakly transitive frames (W, R) and (W', R'). As they are distinct, either $W \neq W'$ or $R \neq R'$. In the first case it is immediate that $(W, \Omega_R) \neq (W', \Omega_{R'})$. In the second case as both R and R' are irreflexive the second floors are not the same, so $W_2 \neq W_2'$ and hence $\Omega_R \neq \Omega_{R'}$.

For surjectivity take an arbitrary minimal topological space (W, Ω), where $\Omega = \{W, \varnothing, W_0\}$ for some subset $W_0 \subseteq W$. Take the frame (W, R), where $R = (W_0 \times W_0 - \{(w, w) | w \in W_0\}) \cup (-W_0 \times -W_0 - \{(w, w) | w \in -W_0\}) \cup \{(w, w') | w \in -W_0, w' \in W_0\}$. In words every two distinct points are related in W_0 by R and the same in the complement $-W_0 = W - W_0$, besides every point from the $-W_0$ is related to every point from W_0. what we get is the rooted two step relation which is weakly transitive, with the second floor equal to W_0. As we didn't allow wRw for any point $w \in W$, the relation R is also irreflexive.

Now we will give the definition of a derived set (or set of accumulation points) of a set in a topological space. This definition is needed to give the semantics of modal formulas in arbitrary topological spaces.

Definition 4.015 *Given a topological space (W, Ω) and a set $A \subseteq W$ we will say that $w \in W$ is an* accumulation point *of A if for every neighborhood U_w of w the following holds: $U_w \cap A - \{w\} \neq \varnothing$. The set of all accumulation points of A will be denoted by $der(A)$ and will be called the* derived *set of A.*

Derived sets serve to give the semantics of the diamond modality in an arbitrary topological space. Below we give the definition of satisfaction in a derived set topological semantics of modal logic.

Definition 4.016 *A topological model (W, Ω, V) is a triple, where (W, Ω) is a topological space and $V : Prop \to P(W)$ is a valuation function. Satisfaction of a modal formula in a topological model (W, Ω, V) at a point $w \in W$ is defined by:*

$$w \Vdash p \quad \textit{iff} \ \ w \in V(p),$$
$$w \Vdash \Diamond p \quad \textit{iff} \ \ w \in der(V(p)),$$

boolean cases are standard. Validity in a frame and class of frames of a formula is defined in a standard way.

Fact 4.017 *Let (W, R) be a finite, weakly transitive and irreflexive frame and let (W, Ω_R) be its Allexandrof space. For every modal formula α the following holds:*

$$(W, R) \Vdash \alpha \ \textit{iff} \ (W, \Omega_R) \Vdash \alpha.$$

Note that here $\Vdash$ on the left hand side denotes the validity in Kripke frames while on the right hand side it denotes the validity in topological frames in derived set semantics.

Theorem 4.018 *The modal logic $wK4F$ is sound and complete with respect to the class of all minimal topological spaces.*

Proof. Proof. Soundness can be checked directly so we do not prove it here. For completeness assume $\nvdash \phi$. By theorem 2.26 there exists a finite, one-step, weak-transitive frame (W, R) which falsifies ϕ. Assume that $Ch(W, R) = (i_1, r_1, i_2, r_2)$.

By theorems 3.011, 3.012 and 3.013 there exists a one-step, weak-transitive frame (W', R') such that R' is irreflexive and (W, R) is p-morphic image of (W', R'). For example such a frame could be $(W', R') = Ch^{-1}(i_1 + 2 \times r_1, 0, i_2 + 2 \times r_2, 0)$. The surjection immediately yelds that $(W', R') \nVdash \phi$. Now the result immediately follows from theorem 4.014, and the fact 4.017.

5 Conclusions

We have characterised the logic $wK4F$ and established its relation to minimal topological spaces. The logic is closely related to the model logic $S4F$ that has been shown to capture several important kinds of knowledge representation systems and, in its non-monotonic version, can be viewed as a logic of minimal knowledge. In future work we plan to study the non-monotonic version of $wK4F$ and its relation to $S4F$ in more detail. It may also be interesting to consider multi-model versions and their suitability for modeling common knowledge.

References

1. Fagin, R., Halpern J.Y., Moses Y., and Vardi M.Y. Reasoning about Knowledge. Published by MIT Press, 1995.
2. Segerberg, K. An Essay in Classical Modal Logic. volume 13 of Filosofiska Studier. Uppsala: Filosofiska Foreningen och Filosofiska Institutionen vid Uppsala Universitet.
3. Truszczynski, M. Embedding Default Logic into Modal Nonmonotonic Logics. LPNMR 1991, 151-165.
4. Truszczynski, M.,Schwarz G. Minimal Knowledge Problem: A New Approach. Artif. Intell. 67(1), 113-141, 1994.
5. Gelfond M., Lifschitz V. Classical Negation in Logic Programs and Disjunctive Databases. New Generation Computing, vol. 9, 365-385, 1991.
6. Reiter R. A logic for default reasoning. Artificial Intelligence, 13:81-132, 1980.
7. M. Gelfond, V. Lifschitz, H. Przy-musinska, and M. Truszcynski. Disjunctive defaults. In Second International Conference on Principles of Knowledge Representation and Reasoning, KR '91, Cambridge, MA, 1991.
8. Lin F., Shoham Y. Epistemic semantics for fixed-points nonmonotonic logics. In Rohit Parikh, editor. Theoretical Aspects of Reasoning about Knowledge: Proc, of the Third Conf, pages 111-120, 1990.
9. Lin F., Shoham Y. Epistemic semantics for fixed-points nonmonotonic logics. In Rohit Parikh, editor. Theoretical Aspects of Reasoning about Knowledge: Proc, of the Third Conf, pages 111-120, 1990.
10. Lifschitz V. Minimal Belief and Negation as Failure. Artificial Intelligence, vol. 70, 53-72, 1994.
11. Truszczynski, M. The Modal Logic $S4F$, the Default Logic, and the Logic Here-and-There. In Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI 2007). AAAI Press, 2007.
12. Cabalar, P., and Lorenzo, D. New Insights on the Intuitionistic Interpretation of Default Logic. In R. López de Mántaras and L. Saitta (eds), ECAI 2004, IOS Press 2004, 798–802.
13. Esakia, L. Weak transitivity - restitution. Logical Studies 2001, vol 8, 244-255.

Argumentation-based Distributed Induction

Santiago Ontañón[1] and Enric Plaza[2]

[1] CCL, Cognitive Computing Lab,
Georgia Institute of Technology, Atlanta, GA 303322/0280,
santi@cc.gatech.edu
[2] IIIA-CSIC, Artificial Intelligence Research Institute
Campus UAB, 08193 Bellaterra, Catalonia (Spain),
enric@iiia.csic.es

Abstract. Argumentation can be used by a group of agents to discuss about the validity of hypotheses. In this paper we propose an argumentation-based framework for distributed induction, where two agents learn separately from individual training sets, and then engage in an argumentation process in order to converge to a common hypothesis about the data. The result is a distributed induction strategy in which the agents minimize the set of examples that they have to share in order to converge to a common hypothesis. The proposed strategy works for any induction algorithm which expresses the hypothesis as a disjunction of rules. We show that the strategy converges to a hypothesis indistinguishable in training set accuracy from that learned by a centralized strategy.

Keywords: Distributed Induction, Argumentation, Classification.

1 Introduction

Distributed induction is the problem of learning a hypothesis or model (such as a set of rules, or a decision true) from data when the data is distributed among different sources. Some real-life domains involve such forms of distributed data, where data cannot be centralized due to one or several of the following reasons: storage size, bandwidth, privacy, or management issues. Storage size and bandwidth are less and less a problem nowadays, however in large data sets they might still be an issue. In this paper we will propose a framework in which agents will use a limited form of argumentation in order to arrive to a model of all the data while minimizing the communication, and specially minimizing the amount of examples exchanged, and ensuring that the hypothesis found is exactly as good as if centralized induction with all the data was used.

Argumentation frameworks can be used in multi-agent systems for different purposes such as joint deliberation, persuasion, negotiation, and conflict resolution [12]. In a previous work [8] we have shown how argumentation can be used by agents that use lazy learning techniques. In this paper we will introduce a framework where agents use argumentation to argue about hypotheses. In this framework, agents will generate hypotheses locally, and then argue about

them until both agents agree. During the argumentation process, agents might exchange a small number of examples.

Formalizing agent communication as argumentation allows the distributed induction strategies to abstract away from the induction algorithm used by the agents. Thus, all the strategies presented in this paper can work with any induction algorithm that satisfies certain requirements. In particular, we require that the hypotheses learnt can be expressed as a disjunction of independent rules. So, for instance, algorithms such as FOIL [11], or ID3 [10] (since a tree can be easily flattened into a set of rules), or other rule learners can be used. Algorithms such as CN2 [4] that learn an *ordered* set of rules also fit in this framework, but rules require some preprocessing to remove the dependencies that the ordering introduces (as elaborated in Section 2). Moreover, the framework is also agnostic in regards to the representation formalism, in the experiments section we will show results with both relational as well as flat feature-vector representations.

The remainder of this paper is organized as follows. Section 2 presents our multi-agent learning framework, including our formalism for argumentation. Section 3 presents two strategies for distributed induction based on argumentation, and Section 4 empirically evaluates them, comparing them to other distributed induction strategies in the literature. Section 5 provides a quick overview of the related work, and finally the paper closes with conclusions and future work.

2 A Framework for Multi-Agent Learning

Let A_1 and A_2 be two agents who are completely autonomous and have access only to their individual and private training sets T_1, and T_2. A training set $T_i = \{e_1, ..., e_n\}$ is a collection of examples. Agents are able to individually apply induction in order to learn a hypothesis (or model) of the data and solve problems using the induced model, but they can also collaborate with other agents for both induction and problem solving. In the rest of this paper we will use the terms model and hypothesis indistinguishably.

2.1 Examples, Hypotheses and Rules

Examples, *hypotheses* and *rules* are the three key concepts of the learning framework proposed in this paper.

We will restrict ourselves to analytical tasks, i.e. tasks, such as classification, where the solution of a problem is achieved by selecting a solution class from an enumerated set of solution classes. In the following we will note the set of all the solution classes by $\mathcal{S} = \{S_1, ..., S_K\}$. Therefore, an *example* $e = \langle P, S \rangle$ is a pair containing a problem P and a solution class $S \in \mathcal{S}$. In the remainder of this paper, we will use the dot notation to refer to elements inside a tuple; e.g., to refer to the solution class of an example e, we will write $e.S$.

Our framework is restricted to hypotheses H that can be represented as a disjunctive set of *rules*: $H = \{r_1, ..., r_m\}$. A rule $r = \langle H, S \rangle$ is composed of a *body* $r.H$, and a *solution*, $r.S$. When a problem P matches the body $r.H$

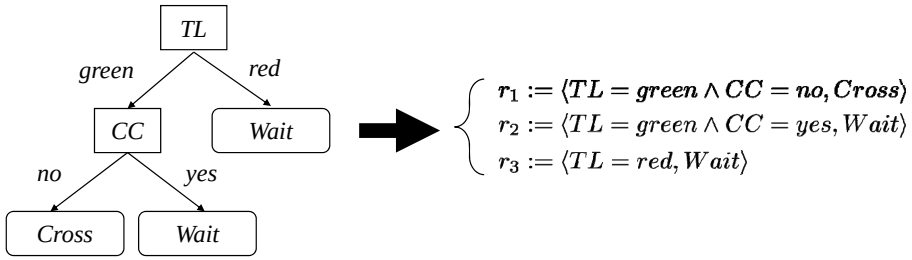


Fig. 1. Conversion of a decision tree to a set of rules.

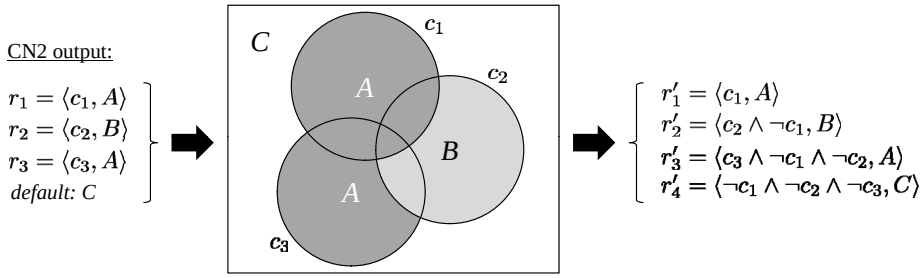


Fig. 2. Postprocessing of the rules generated by CN2 in order to remove the order dependencies, and thus fit in our argumentation framework.

of a particular rule r, the rule predicts that the solution to the problem P is $r.S$. When a problem matches the body of a rule $r.H$, we say that the body *subsumes* the problem: $r.H \sqsubseteq P$. A large number of induction algorithms can generate hypothesis that can be represented using this formalism. In particular, in our experimental section we have selected to use INDIE [1], which is a heuristic relational inductive learner, ID3, and CN2, but other algorithms could have been used such as AQR [6], or FOIL [11].

The framework introduced in this paper does not specify which induction algorithm or representation formalism agents use. In principle, any induction algorithm could be used, and any data representation (propositional, relational, or any other) could be used. The restriction on the hypothesis representation is imposed because agents will argue about each of these rules independently.

To illustrate how different induction algorithms can represent their hypothesis using our formalism, Figure 1 shows the equivalence between a decision tree and a set of rules. In the example, a simple decision tree with only two features (TL and CC) is shown, and three rules equivalent to the tree are constructed.

Further, as we mentioned earlier, when using algorithms such as CN2, that produce an ordered set of rules, the rules produced have to be postprocessed in order to remove the order relationship among them. The left hand side of Figure 2 shows a set of three rules generated by CN2 (plus the default class assigned by

CN2 when no rule covers a problem). The center of Figure 2 shows a graphical representation of the way these rules partition the problem space among the three different solution classes A, B, and C (the three circles represent the subset of problems that are subsumed by each of the three conditions in the body of the rules: c_1, c_2 and c_3). Notice for instance that rule r_2 states that all the problems that are subsumed by c_2 have solution B. However, r_2 is only considered if r_1 is not fired. Therefore, that rule is postprocessed and converted into rule r'_2, which states that all examples that are subsumed by c_2, but not by c_1 have solution B. In general, a rule is postprocessed by adding the negations of all the previous rules to its body. Finally, the default solution computed by CN2 is converted also into a rule containing the conjunction of the negation of the body of all the rules generated by CN2. The result of this process is a set of independent rules, which can be used in our framework.

As illustrated by the previous two examples, a large collection of induction algorithms can represent their hypotheses in the form of rules.

2.2 Arguments and Counterarguments

In order to use argumentation, two elements must be defined: the *argument language* (that defines the set of arguments that can be generated), and a *preference relation* (that determines which arguments are stronger than others). In our framework, the argument language is composed of two kinds of arguments:

- A *rule argument* $\alpha = \langle A, r \rangle$, is an argument generated by an agent $\alpha.A$ stating that the rule $\alpha.r$ is true.
- A *counterexample argument* $\beta = \langle A, e, \alpha \rangle$, is an argument generated by an agent $\beta.A$ stating that a particular argument $\beta.\alpha$ is incorrect, because the example $\beta.e$ is a counterexample of such argument.

To define the relation among arguments, we have to take into account all the possible different situations that can arise while comparing two arguments consisting of rules or examples. Figure 3 shows all these situations. The top row of Figure 3 considers all the possible comparisons of two rule arguments, r_1 and r_2 such that $r_1.S = r_2.S$. Only three situations might arise: a) r_1 and r_2 are totally unrelated, b) the sets of problems covered by r_1 and r_2 have a non empty intersection, and c) one is more general than the other. The middle row of Figure 3 considers all the possible comparisons of two rule arguments, r_1 and r_2 but this time $r_1.S \neq r_2.S$. The same three situations arise (unrelated, non-empty intersection, and one more general than another). Notice that in the non-empty intersection situation we also require that no rule is more general than another (we don't include the extra restriction in the figure for clarity). Thus, when comparing any two rule arguments, only 6 situations might arise. Situations a), b), c) and d) represent rule arguments that are *compatible*, whereas situations e) and f) represent *conflicting* arguments. Moreover situation c) is a special situation and we say that r_1 *subsumes* r_2.

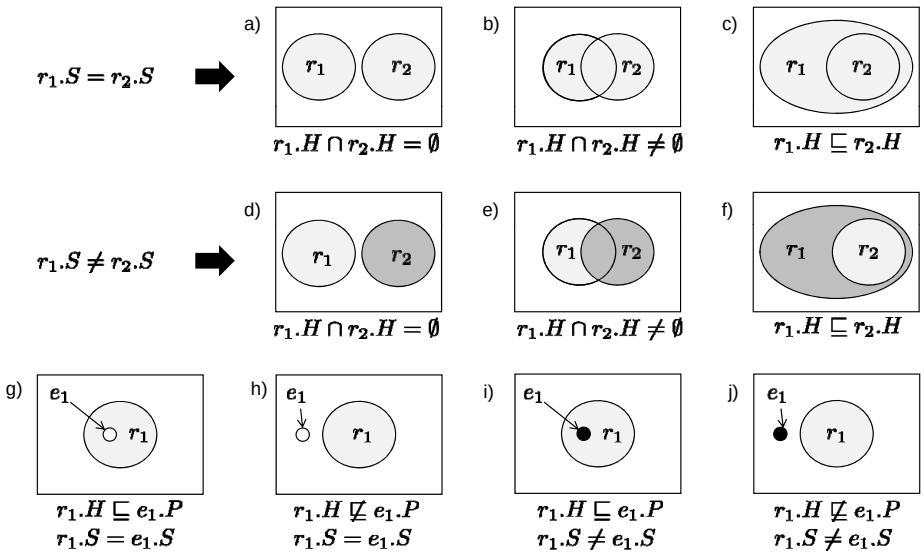


Fig. 3. All the possible different situations that can arise while comparing two arguments consisting of rules or counterexamples.

The third row of Figure 3 shows all the possible situations that arise when comparing a rule argument with a counterexample argument: g) both the counterexample and the rule support the same class (in which case the counterexample is not such, and both arguments endorse each other), h) in which the counterexample, although supporting the same class, is not covered by the rule, i) where the counterexample supports a different solution than the rule, and the rule covers the counterexample, j) in which the counterexample, although supporting a different class, is not covered by the rule. In our framework, we assume that a counterexample cannot be defeated, and thus only the rule arguments can be defeated. Out of the four situations, the counterexample argument only defeats the rule in situation i), in all the other situations, they are compatible. The counterexample in situation i) is called a *defeating counterexample* of r_1.

Using these two types of arguments and the *compatible*, *conflicting*, *subsumed*, and *defeated* relations among arguments, next section introduces two different distributed induction strategies.

3 Argumentation-based Distributed Induction

In this section we will present two strategies, ADI (Argumentation-based Distributed Induction) and RADI (Reduced Argumentation-based Distributed Induction), based on argumentation for distributed induction. Since the strategies involve communication among agents, they will be presented as communication

protocols. Both strategies are based on the same idea, and share the same high level structure.

1. A_1 and A_2 use induction locally with their respective training sets, T_1 and T_2, and obtain initial hypotheses H_1 and H_2 respectively.
2. A_1 and A_2 argue about H_1, obtaining a new H_1^* derived from H_1 that is consistent with both A_1 and A_2's data.
3. A_1 and A_2 argue about H_2, obtaining a new H_2^* derived from H_2 that is consistent with both A_1 and A_2's data.
4. A_1 and A_2 obtain a final hypothesis $H^* = H_1^* \cup H_2^*$. Remove all the rules that are subsumed by any other rule (situation c) in Figure 3).

Basically, the intuitive idea is the following. In step 1 both agents perform induction individually. Then in steps 2 and 3 (which are symmetric, and can actually be performed in parallel), the agents use argumentation to refine the individually obtained hypotheses and make them compatible with the data known by both agents. Finally, when both hypothesis are compatible, a final global hypothesis H^* is obtained by just computing the union of all the rules learned by both agents, removing all the rules that are subsumed by some other rule (since those would be redundant). Notice that, unless the induction algorithms are not able to learn rules with 100% accuracy in the training set, there should not be any conflicting rules in H^* (at least not conflicting in the classification of the examples of the training set). However, there might be rules that conflict in the classification of problems outside of the training set. If the learning algorithm computes confidence levels for rules, those can be used to arbitrate, otherwise random arbitration can be used. ADI and RADI only differ in the way steps 2 and 3 are performed.

Step 2 in ADI works as follows

1. Let $H_1^0 = H_1$, and $t = 0$.
2. If there is any rule $r \in H_1^t$ that still has not been *accepted* by A_2, then send the argument $\alpha = \langle A_1, r \rangle$ to A_2. Otherwise, if all the rules in H_1^t have been accepted the protocol goes to step 5.
3. A_2 analyzes $\alpha.r$ and tries to find a counterexample that defeats it (situation i) in Figure 3). If A_2 can find such counterexample e, then A_2 sends the counterargument $\beta = \langle A_2, e, \alpha \rangle$ to A_1. Otherwise, r is accepted and the protocol goes to step 2 again.
4. When A_1 receives a counterexample β, it appends $\beta.e$ to its training set T_1, and updates its hypothesis. If the induction algorithm of A_1 is not incremental, then A_1 can simply use induction from scratch with the new extended set of examples that includes e. A_1 updates the hypothesis obtaining H_1^{t+1}. The protocol goes to step 2 again, and $t = t + 1$.
5. The protocol returns H_1^t.

The main idea is that A_1 will generate hypotheses according to his local training set T_1, and A_2 evaluates them, trying to generate counterarguments to the hypotheses that do not agree with his own local data T_2. Step 3 in ADI is

just the reversed, where it is A_2 that generates hypotheses, and A_1 that tries to rebut them with counterexamples. One characteristic of ADI is that at each step, only one counterexample is exchanged.

If the induction algorithm of A_1 and A_2 was capable of achieving 100% accuracy if it was given the complete collection of examples that both A_1 and A_2 have, then the protocol always ends up converging to a hypothesis that also has 100% accuracy in both T_1 and T_2. Moreover, in order to prevent infinite iterations in the case of noisy data, the agents are not allowed to send the same counterexample twice during the protocol. This ensures that the protocol will eventually end.

The second strategy, RADI, improves over ADI in trying to minimize the number of times the hypothesis has to be updated while keeping the number of counterexamples exchanged low. Step 2 in RADI works as follows:

1. Let $H_1^0 = H_1$, and $t = 0$.
2. Let $R^t \subseteq H_1^t$ be the set of rules in the hypothesis of A_1 not yet *accepted* by A_2. If such set is empty, then the protocol goes to step 5. Otherwise, A_1 sends the set of arguments $\mathcal{R}^t = \{\alpha = \langle A_1, r\rangle | r \in R^t\}$ to A_2.
3. For each $\alpha \in \mathcal{R}^t$, A_2 computes the set of examples C_α in its training set that are counterexamples that defeat $\alpha.r$: $C_\alpha = \{e \in T_2 | \alpha.r.H \sqsubseteq e.P \wedge \alpha.r.S \neq e.S\}$. For each argument $\alpha \in \mathcal{R}^t$ such that $C_\alpha = \emptyset$, $\alpha.r$ is accepted by A_2. Let $I^t \subseteq \mathcal{R}^t$ be the subset of arguments for which A_2 could find defeating counterexamples. A_2 computes the minimum set of counterexamples B^t such that $\forall_{\alpha \in I^t} C_\alpha \cap B^t \neq \emptyset$. That is the minimum subset of examples that can defeat all arguments in I^t. A_2 sends the set of counterexample arguments $\mathcal{B}^t$ consisting of a counterexample argument $\beta = \langle A_2, e, \alpha\rangle$ for each pair e, α such that $e \in B^t$, $\alpha \in \mathcal{R}^t$, and β defeats α.
4. When A_1 receives a set counterexample arguments $\mathcal{B}^t$, it appends all the examples on them to its training set T_1, and updates its hypothesis. If the induction algorithm of A_1 is not incremental, then A_1 can simply use induction from scratch with the new extended set of examples. A_1 updates the hypothesis obtaining H_1^{t+1}. The protocol goes to step 2 again, and $t = t + 1$.
5. The protocol returns H_1^t.

The idea behind RADI is that an example can be a defeating counterexample of more than one rule at the same time, thus, by selecting the minimum set of counterexamples, the number of examples exchanged is reduced. Also, by sending all the counterexample arguments at once, the number of times the hypothesis has to be updated is also reduced. This results in an efficient strategy for distributed induction that minimizes the number of examples being exchanged, and that can be used with any induction algorithm. Notice that finding such minimum subset of counterexamples is NP, however approximate methods to compute such minimum subset can be easily defined.

Figure 4 illustrates the first cycle of the RADI protocol. In the figure, agent A_1 has learnt a hypothesis H_1^0 from its original training set. A_1 sends the set $\mathcal{R}^0$ of rule arguments to A_2. In the middle part of Figure 4, we can see the

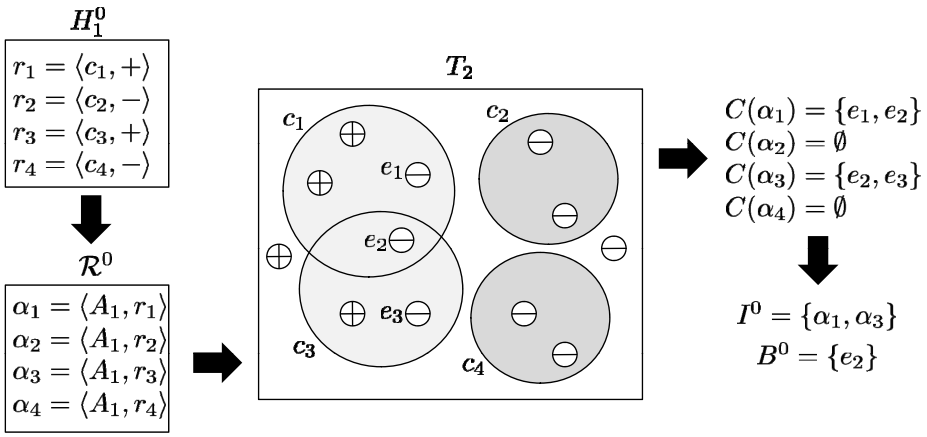


Fig. 4. An illustration of one step in the RADI strategy.

training set of A_2 (T_2). In this example, there are only two classes, + and -. A_2 evaluates all the arguments in $\mathcal{R}^0$ with his training set T_2. In particular, in this example, A_2 finds counterexamples for two of the arguments, namely α_1 and α_2. Thus, the set $B^0 = \{\alpha_1, \alpha_2\}$. Then, A_2 constructs the set B^0 consisting of the minimum set of examples that contain counterexamples for all the arguments in I^0. In this case, there is a particular example, e_2, which is a counterexample of both arguments, so e_2 is enough to contradict both. Therefore, A_2 will construct the set of counterarguments $\mathcal{B}^0 = \{\langle A_2, e_2, \alpha_1 \rangle, \langle A_2, e_2, \alpha_3 \rangle\}$, and send it to A_1, which will update his hypothesis by appending e_2 to its training set T_1, and will generate a new updated hypothesis H_1^1, with which the next round of the protocol will start.

As explained in Section 6, it is part of our future work to investigate how the inclusion of additional types of counterarguments, such as rule counterarguments, can further reduce the amount of information exchanged during the distributed induction process.

4 Experimental Evaluation

In order to evaluate our approach, we tested the distributed induction strategies in four different data sets: three propositional data sets from the Irvine machine learning repository (soybean, zoology, cars), and a complex relational data set (sponges). Moreover, we tested it using three different induction algorithms: ID3, CN2 and INDIE (a relational inductive learner [1]). We also compared the results against centralized induction and also three other distributed induction strategies: individual (where agents just do induction individually), union (where agents do induction individually, and then they put together all the rules they learn into one common hypothesis), and DAGGER [5] (the only other distributed induction technique independent of the learning algorithm to the best of our

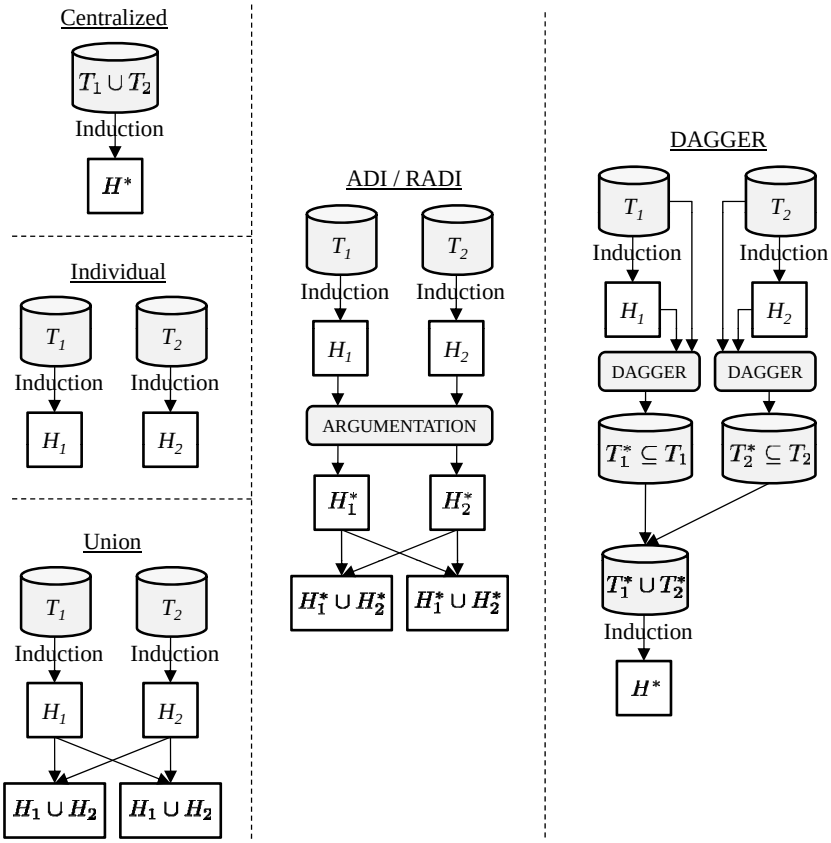


Fig. 5. An illustration of all the different distributed induction strategies evaluated in our experiments.

knowledge, see Section 5 for a brief explanation of DAGGER). Figure 5 presents a visual overview of the different strategies used in our evaluation. We evaluated convergence, time, number of examples exchanged, number of rules exchanged, number of induction calls, and both training and test set accuracy. All the results presented are the average of 10 fold cross validation runs.

Since sponge is a relational data set (introduced by Armengol and Plaza [1]) it has to be converted to propositional so that ID3 and CN2 can use it. In the sponge data set, examples are represented as trees, and the size of the trees varies greatly from an example to another. In order to convert it to a propositional representation, we computed the set of all possible different branches that the examples have, and each one is converted to a feature (70 different features are defined in this way). Each example consists of about 30 to 50 features each, so there is a large amount of missing values in the resulting propositional representation. Thus, both ID3 and CN2 have troubles learning in this domain. CN2

Table 1. Training and test accuracy measurements of different distributed induction strategies combined with different induction algorithms.

	Training				Test			
	Soybean	Zoology	Cars	Sponges	Soybean	Zoology	Cars	Sponges
ID3-ADI	100.00	100.00	100.00	99.70	**88.50**	**99.00**	88.95	58.21
ID3-RADI	100.00	100.00	100.00	99.74	87.67	**99.00**	89.24	58.21
ID3-centralized	100.00	100.00	100.00	99.44	85.00	**99.00**	88.95	58.57
ID3-individual	85.67	93.85	93.84	80.20	76.50	90.00	86.84	55.54
ID3-union	90.25	94.73	97.73	94.05	81.00	94.00	**90.99**	60.36
ID3-DAGGER	99.57	100.00	76.36	99.76	80.67	92.50	68.95	**62.50**
CN2-ADI	100.00	100.00	100.00	100.00	**84.90**	**93.50**	**80.61**	**79.11**
CN2-RADI	100.00	100.00	100.00	100.00	**84.66**	**93.50**	80.17	**78.93**
CN2-centralized	100.00	100.00	100.00	100.00	**84.66**	94.00	**80.64**	**78.57**
CN2-individual	87.82	94.62	89.90	88.29	77.83	87.50	**80.84**	74.46
CN2-union	54.91	91.65	80.41	70.71	63.66	86.00	80.00	68.20
CN2-DAGGER	99.49	99.65	95.86	99.88	79.33	92.50	75.34	**78.93**
INDIE-ADI	99.64	100.00	100.00	100.00	**84.33**	93.00	**91.25**	**95.89**
INDIE-RADI	99.64	100.00	100.00	100.00	**84.50**	**94.00**	**91.37**	94.11
INDIE-centralized	99.64	100.00	100.00	100.00	83.00	**94.00**	81.80	95.00
INDIE-individual	89.21	94.07	93.93	96.45	77.50	85.50	87.76	54.11
INDIE-union	91.44	96.48	97.42	97.90	78.00	90.00	**91.80**	94.29
INDIE-DAGGER	-	-	-	-	-	-	-	-

does, in fact, a better job, but ID3 achieves a very low classification accuracy. Additionally, since the basic ID3 cannot handle missing values, all missing values where considered to have the special value "missing" when the data set was used by ID3. For CN2, a beam size of 3 was used in all the experiments.

Table 1 presents the classification accuracy (both measured in training set and in test set). We ran each combination of induction algorithm (ID3, CN2, INDIE) with distributed induction strategy (centralized, individual, union, DAGGER, ADI and RADI) with all the data sets (except the combination of INDIE-DAGGER, that is not possible, since DAGGER assumes propositional data sets, and INDIE requires them in relational form). In each experimental run the data set was split in two sets, a training set containing 90% of the examples, and a test set containing 10% of the examples. The training set was further split among two agents (except in the case of the centralized strategy, where there was only one agent). Accuracy is measured in the original training set (with 90% of the examples), and also in the remaining 10%, that forms the test set. The left hand side of Table 1 shows accuracy in the training set, and the right hand side shows accuracy in the test set.

Looking at the training accuracy, the first thing that the experimental results confirm is that the hypotheses learnt by ADI and RADI are indistinguishable in training set accuracy from the one learnt by using centralized induction. Achieving a 100% accuracy all the times where centralized induction also does.

Table 2. Time (in seconds) required to complete the induction process per agent, number of examples shared per agent (as a percentage of the number of examples owned by an agent), number of rules sent per agent, and number of times the base induction algorithm had to be invoked per agent (notice that in the "centralized" case, there is only one agent). All the results are average over all the induction algorithms and all the data sets.

	time	Examples shared	Rules sent	Induction calls
centralized	2.8	100.00%	0.00	1.00
individual	1.5	0.00%	0.00	1.00
union	1.5	0.00%	67.63	1.00
DAGGER	3.5	68.56%	64.75	1.50
ADI	155.4	19.04%	3748.70	58.90
RADI	18.2	21.52%	679.34	5.77

When agents perform individual induction, of course, training accuracy diminishes (since agents only learn with 50% of the data in the training set), agents using the union strategy improve their accuracy, but still it is not guaranteed to be as good as that of centralized accuracy (and in the case of CN2, where the order of the rules matter, the accuracy drops drastically). DAGGER shows good accuracy (although not guaranteeing that of centralized induction).

Analyzing test set accuracy, we observe that, except in a few cases where DAGGER achieves higher accuracy (and one where surprisingly union does [3]), ADI and RADI achieve same or higher accuracy than the centralized approach. Table 1 shows the highest results for each induction algorithm in boldface (when the difference was not statistically significant, more than one result is highlighted). The explanation is that when agents use ADI or RADI, two different hypothesis of the data are learnt (one per agent), and, after inconsistencies are fixed, they are merged. Therefore, the resulting hypothesis has potentially several rules that cover the same examples, but that were derived from different training sets (thus having different biases). This, alleviates overfitting, and thus increases classification accuracy in unseen problems. The effect achieved is similar to that of ensemble methods, but with the advantage of having a single hypothesis.

Another effect that can be seen is that ID3 and CN2 cannot properly handle the complexity of the sponges data set, since, although they can achieve high training set accuracy, the rules they learn do not generalize and achieve very low test set accuracy. INDIE, however, being a relational learner, can handle sponges in its native representation formalism, and thus learn much more general rules, that generalize properly, achieving high test set accuracy.

Classification accuracy, however, is only one side of the coin. Table 2 shows the amount of time used by each of the different distributed induction strategies

[3] We repeated that experiment several times, with identical result, we are still in the process of analyzing why union achieves such good result in the cars data set with ID3.

per agent (averaged over all the data sets and induction algorithms), also the percentage of the examples owned by each agent that had to be shared, the number of rules exchanged, and also the number of times that the agents had to call the base induction algorithm. Notice, that time is dominated by the slower learning algorithm (CN2) and the most complex data set (sponges), the fastest algorithm (ID3) required less than a tenth of a second for any strategy except ADI and RADI (where it still required less than a second for any data set). Table 2 shows that ADI and RADI, are the most computationally expensive strategies, ADI taking 155.4 seconds and RADI 18.2, while centralized accuracy required only 2.8 seconds. Moreover, most of the time consumed by ADI and RADI corresponds to invocations to the base induction algorithm after receiving new examples. If an incremental induction algorithm such as ID5R or ITI [15] the amount of time consumed would be reduced drastically.

Table 2 shows that among all the distributed induction strategies, DAGGER is the one that requires exchanging the highest percentage of examples, 68.56%, while ADI and RADI exchange only 19.04% and 21.52% respectively. The union strategy, of course, does not force agents to exchange any example. However, ADI and RADI require the exchange of a large amount of rules, where as other strategies, such as DAGGER, or union only require sharing the final hypothesis. While ADI shares slightly a lower amount of examples, RADI requires only a tenth of the time, a fifth of the rules, and a tenth of the number of induction calls.

Summarizing the results, we can conclude that different distributed induction strategies have different strengths and weaknesses. Performing centralized induction has the problem of having to share all the examples, but achieves a high accuracy at the minimum computational cost. Next in line is DAGGER, which forces the agents to share most of their examples, but achieves a high accuracy also (although not guaranteed to be as high as centralized). On the other extreme, we have the individual and union strategies, that have the minimum computational cost, zero example exchange, but also the lowest classification accuracies. ADI and RADI sit in the middle, requiring the agents to share a small percentage of examples (around 20%), while ensuring the same or higher classification accuracy than centralized induction (especially in the test set, where they hypotheses learnt by ADI or RADI have less overfitting). However, ADI and RADI have a higher computational cost. In between ADI and RADI, we can conclude that RADI is the most well balanced strategy, since it requires about a tenth of the computational cost, while only sharing a very small number of additional examples.

Moreover, notice that nothing stops ADI or RADI from working even if each of the agents had a different base induction algorithm.

5 Related Work

Two areas of work are related to ours, namely distributed induction and incremental learning. Distributed induction has been attempted mainly from four

different approaches: computing statistics in a distributed fashion and then aggregating, sharing example, sharing hypotheses or viewing induction as search and distributing the search process. Let us present examples of each of the approaches.

Caragea et al. [3] present a framework for induction from a set of distributed sources based on computing a collection of statistics locally in each of the sources, and then aggregating them to learn a model. They prove that some learning algorithms, such as ID3, can be distributed in this way while still guaranteeing finding the same exact tree that would be found if all the data were centralized. Their framework restricts to feature value representations. The main difference with our work is that in their framework they assume a single agent trying to learn from scratch from a collection of distributed sources, while in our framework we assume a multi-agent system with agents that already have an initial hypothesis and improve it by interacting with other agents. A similar framework was presented by Bhatnagar and Srinivasan [2], but where they allow each agent to have a completely different set of attributes, as long as all the tables owned by the agents can be put together using a *join* data base operation if they were copied in a centralized repository. Another difference of both these frameworks with ours is that both attempt at providing a framework for defining distributed induction algorithms. So, using those frameworks, algorithms such as ID3 or CN2 have to be adapted to work in a distributed fashion. In contrast, our research focuses on finding distributed induction strategies that can be built around standard induction algorithms without modifying them.

A different approach is DAGGER, presented by Davies [5] (and used in our experiments for comparison purposes). Davies proposes to perform distributed induction by selecting a reduced set of informative examples from each of the distributed sources, and then perform centralized induction with the union of the reduced sets of examples. Davies' method had the goal of being an alternative to voting mechanisms for aggregating hypothesis, which is a different goal than the work presented in this paper. Thus Davies' approach is a one shot approach that does not ensure preserving classification accuracy, while our strategies do.

Another approach to distributed induction is that of Shaw and Sikora [14], where they propose to learn individual models in each of the sources, and then combine them by using a genetic algorithm that uses specialized mutation and crossover operators for being able to merge the hypothesis. The goal with Shaw and Sikora's approach is just to distribute the induction task among agents, so that it becomes more efficient and parallelizable. Our goal is not to make the induction process more efficient, but to allow groups of agents to perform distributed induction by using their own induction algorithm, and ensuring high quality of the hypotheses found.

Another example of distributing induction for efficiency is that of distributing the search process of finding rules among a series of distributed processors. Provost and Hennessy [9] propose to perform distributed search for rule learning, where each individual processor only searches with a subset of the data and proposes each candidate rule to the rest for verification.

Also related is the work on incremental learning, such as the incremental versions of the ID3 algorithm ID4 and $\widehat{\text{ID4}}$ by Schlimmer and Fisher [13], or ID5R and ITI by Utgoff [15], which allow learning a decision tree from a set of examples, and then update it with new examples at a lower cost than learning it from scratch again. These algorithms can be used to increase the efficiency of relearning the hypotheses in the techniques presented in this paper. Experiments to validate this claim are part of our future work.

Finally, the argumentation framework presented in this paper is complementary to that introduced in [8], where an argumentation framework for lazy learning called AMAL was presented. The idea behind AMAL is complementary to that of ADI and RADI. While in ADI and RADI agents perform induction in a collaborative fashion, and then they solve problems individually (by using the hypothesis learnt collaboratively), in AMAL, agents learn separately, and only collaborate during problem solving. Thus, AMAL is an argumentation model of multi-agent learning based on "solution merging", where as ADI and RADI are based on "hypothesis merging".

6 Conclusions and Future Work

In this paper we have introduced two different distributed induction strategies, ADI and RADI, that can be used on top of any induction algorithm capable of learning hypotheses that can be represented using an independent set of rules. ADI and RADI ensure that the hypothesis learnt will be undistinguishable in terms of training set accuracy from that produced by the base induction algorithm when learning from all the data. The main idea behind ADI and RADI is to let each agent perform induction individually, then argue about the learnt hypotheses to remove inconsistencies, and finally merge both hypotheses.

Experimental results show that, in addition to achieve the same training set accuracy than a centralized method, ADI and RADI have the advantage of obtaining hypotheses that are less prone to overfitting, and thus achieve higher test set accuracy. Moreover, ADI and RADI require sharing only about 20% of the examples of each agent in order to converge to the common hypothesis. ADI and RADI also require that the agents perform several calls to the base induction algorithm, and thus are better suited to be paired with incremental learning algorithms (but this is not required).

ADI and RADI use counterexamples as the only form of counterargument. However, we plan to investigate more complex argumentation protocols that let agents use rule also as counterarguments. The problem of that, is that the base learning algorithms would have to be modified to be able to take rules into account, in addition to the examples in the training set. This is related to the research in "argument based machine learning" by Možina et al. [7] where they modify the CN2 algorithm to take into account specific rules (arguments) in addition to examples for learning purposes. Our ultimate goal is to design distributed induction strategies that could be paired with any induction algorithm, and get as close as possible to not requiring the exchange of any example, while

converging to the same hypothesis of a centralized learner. Additionally, we want to extend ADI and RADI to work with an arbitrary number of agents, and go beyond classification tasks.

Acknowledgements Support for this work came from the project MID-CBR TIN2006-15140-C03-01.

References

[1] E. Armengol and E. Plaza. Bottom-up induction of feature terms. *Machine Learning Journal*, 41(1):259–294, 2000.

[2] Raj Bhatnagar and Sriram Srinivasan. Pattern discovery in distributed databases. In *AAAI/IAAI*, pages 503–508, 1997.

[3] Doina Caragea, Adrian Silvescu, and Vasant Honavar. Decision tree induction from distributed, heterogeneous, autonomous data sources. In *In Proceedings of the Conference on Intelligent Systems Design and Applications (ISDA 03)*, pages 341–350. Springer Verlag, 2003.

[4] Peter Clark and Tim Niblett. The CN2 induction algorithm. In *Machine Learning*, pages 261–283, 1989.

[5] Winston H. E. Davies. *The Communication of Inductive Inference*. PhD thesis, University of Aberdeen, 2001.

[6] Ryszard S. Michalski, Igor Mozetic, Jiarong Hong, and Nada Lavrac. The multi-purpose incremental learning system aq15 and its testing application to three medical domains. In *AAAI*, pages 1041–1047, 1986.

[7] Martin Možina, Jure Žabkar, and Ivan Bratko. Argument based machine learning. *Artificial Intelligence*, 171(10-15):922–937, 2007.

[8] Santiago Ontañón and Enric Plaza. Learning and joint deliberation through argumentation in multiagent systems. In *AAMAS*, page 159, 2007.

[9] Foster John Provost and Daniel N. Hennessy. Scaling up: Distributed machine learning with cooperation. In *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 74–79. AAAI Press, 1996.

[10] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[11] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[12] Iyad Rahwan, Simon Parsons, and Chris Reed, editors. *Argumentation in Multi-Agent Systems, 4th International Workshop, ArgMAS 2007, Honolulu, HI, USA, May 15, 2007, Revised Selected and Invited Papers*, volume 4946 of *Lecture Notes in Computer Science*. Springer, 2008.

[13] Jeffrey C. Schlimmer and Douglas H. Fisher. A case study of incremental concept induction. In *AAAI*, pages 496–501, 1986.

[14] Michael J. Shaw and Riyaz Sikora. A distributed problem-solving approach to rule induction: Learning in distributed artificial intelligence systems. Technical Report ADA232822, Carnagie-Mellon University, 1990.

[15] Paul E. Utgoff. An improved algorithm for incremental induction of decision trees. Technical Report 94-072, UMass, 1994.