Toward a Modular Architecture of Argumentative Agents to Compose Services*

Maxime Morge[1], Jarred McGinnis[2], Stefano Bromuri[2], Francesca Toni[3], Paolo Mancarella[1], and Kostas Stathis[1]

[1] Dipartimento di Informatica
Università di Pisa
via F. Buonarroti, 2 I-56127 Pisa, Italy
{morge,paolo}@di.unipi.it,
[2] Department of Computer Science
Royal Holloway, University of London
McCrea Building, Egham, Surrey TW20 0EX, UK
{jarred,stefano,kostas}@cs.rhul.ac.uk
[3] Department of Computing
Imperial College London
South Kensington Campus, London SW7 2AZ, UK
ft@doc.ic.ac.uk

Abstract. In this paper, we present a model of agents which use argumentation to select and compose services in open and distributed environments. For this purpose, we propose a modular agent architecture using three main modules, dedicated, respectively, to decision making, communication, and negotiation. We deploy a simple "virtual" travel agent example to illustrate how our agents select and compose services, focusing on the functionalities of the modules within the agents.

1 Introduction

The notion of service provides a useful programming metaphor for components of open and distributed systems. Service-oriented architecture are modular, since each service offers an interface between providers and requestors. This interface allow autonomous agents to provide complex services with added value, based upon the composition of atomic services [2].

Argumentation, simply stated, focuses on interactions where parties plead for and against some conclusion [3]. It provides a powerful framework for interacting agents making decisions, assessing the validity of information they become aware of, or resolving conflicts and differences of opinion. It is an essential ingredient of decision making [4], inter-agent communication [5], and negotiation [6].

In this paper we propose an argumentative agent model and architecture whereby agents can reason and make decisions, and can communicate and negotiate with other agents with the aim of supporting service selection and composition, as envisaged in the ARGUGRID project [4]. The reasoning, communication,

* This paper was already published in a french conference [1].
[4] http://www.argugrid.eu/

and negotiation capabilities of agents can support the selection and integration of services in an open and distributed environment. The user requesting a service, as a result, is only required to specify an abstract description of her needs for this service, possibly with some constraints and preferences about it. Similarly, providers of services are only requested to specify an abstract description of their goals, with constraints and preferences. The task of selecting this service (and its provider), and possibly, in the case of a complex service, integrating different atomic services, is then assigned to the agents.

The main focus and contribution of this paper is to provide an in-depth report on the architecture of our agents, and in particular on its *mind* component. This mind is divided in three modules: (i) the *individual decision making* module, allowing to shift from the goals, preferences, and constraints provided by the user to an abstract representation of the required (integrated) services; (ii) the *social decision making* module, allowing to shift, by means of negotiation, from this abstract representation to a concrete one, in terms of concrete services and their parameters; and (iii) the *social interaction* module, managing the communication given social rules of interaction. The consequence is the orchestration of the services is done at the agent level using argumentations techniques.

The rest of this paper is structured as follows. Section 2 gives an overview of the agent architecture. Section 3 outlines a simple example of service selection, requiring composition. Section 4 describes the individual decision making module. Section 5 provides the social decision making module. Section 6 outlines the social interaction module. Then, Section 7 illustrates the service composition process in the context of our illustrative example, by describing the operation of the various modules in coordination. Section 8 discusses related work and, finally, Section 9 concludes by summarising our proposal and discussing our plans for the future.

2 Architecture

In this section, we outline the mind/body architecture of our agents, focusing on the mind component.

Our agent architecture, pictured in Figure 1, is adapted from the mind/body architecture of PROSOCS agents [7]. The *body* senses what is external to it by using the *Communication Module* (CM), which can access the external world, i.e. the events generated by the *Graphical User Interface* (GUI) [5] as well as the messages coming from other agents. Messages received are then stored in the *Incoming Message Queue* (IMQ). Similarly, the messages that the agent wants to send are stored in the *Outgoing Message Queue* (OMQ). These message queues are accessed by the *mind* which is effectively treated as a process that produces speech acts for other agents and events for the GUI. More importantly, the mind and the body can function as co-routines, thus allowing the reasoning process of the mind to be performed concurrently with the body's (communicative) action execution and sensing of the environment.

[5] The GUI is external to the agent in order to provide agents that are lightweight.

In our architecture, the mind of the agent is divided into three modules, with separate concerns.

The first module is the *Individual Decision Making Module* (IDMM), reasoning about the kind of services which can be provided/requested. The IDMM is supported by the concrete data structures in the *Individual Knowledge Base* (IKB). Concretely, if the agent is looking for services, the IKB may contain the abstract representation of the required services, possibly organised in workflows, expressing some constraints. Decisions are made according to the knowledge and the goals of the user, the alternative types of services, and the preferences over them (cf section 4).

The second module is the *Social Decision Making Module* (SDMM), reasoning about the concrete instances of services which can be provided/requested. The SDMM is supported by the concrete data structures in the *Social Knowledge Base* (SKB). Concretely, if the agent is looking for services, the SKB may contain the concrete representation of individual services or workflows of services, and typically the representation of corresponding service providers. Decisions are made according to this knowledge, the goals of the agent, the alternative concrete services, and preferences over them (cf section 5). Both knowledge bases, IKB and SKB, change as new information comes in from the agent's interactions with the user and other agents.

The third module is the *Social Interaction Module* (SIM), driving the social interactions. The SIM is supported by the concrete data structures in the *Protocol Library* (PL), containing a representation of the interaction protocols. Decisions which are required to conduct the interaction according to the protocols of the SIM are provided by the SDMM. The communication module (CM) handles the low level issues concerning the physical passing of messages between the user, the agent's IDMM and SIM and the multiagent system.

As suggested by [8], agents, which have their own private representations, record their interlocutors' commitments. *Commitments* are data structures which contain propositional and action commitments involving the agent, namely with the agent being either the debtor (called *external commitments*) or the creditor (called *individual commitments*). These data structures are shared by the SIM and the SDMM. While the SIM updates all commitments, the SDMM evaluates the external commitments.

3 Walk-through example

We consider here a simple service composition problem for booking a vacation.

In order to be successful, a booking requires a proper understanding of all relevant aspects. Detailed needs for the booking such as destination, cost, and quality of service need to be taken into account. Moreover, any prior knowledge about the services, such as safety of the transport service and features/location of accommodation, are also of vital importance.

The agent is responsible for suggesting a suitable complex service, based on the explicit users'needs and the agent's knowledge. The main goal (g_0), that con-

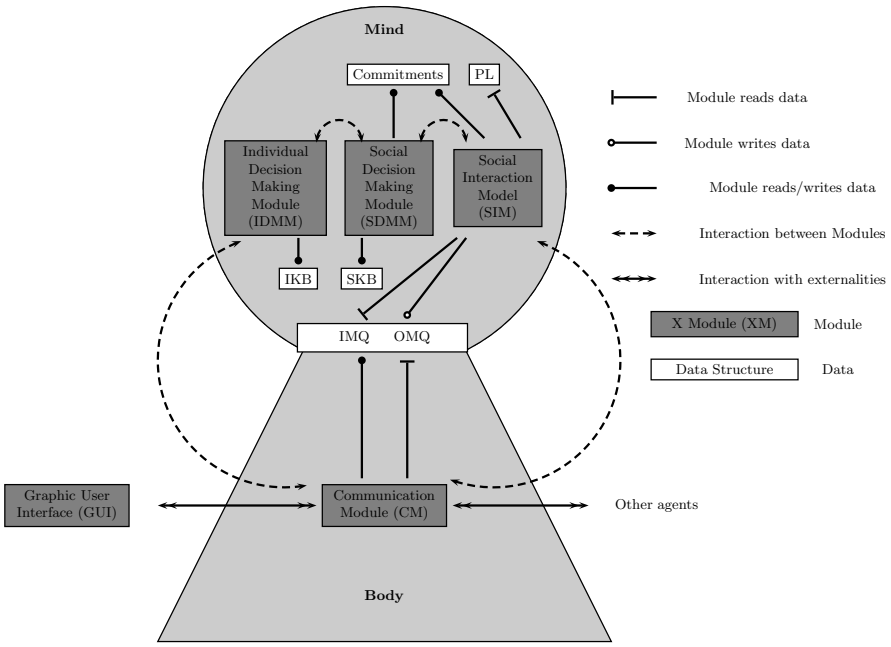

Fig. 1. The modular architecture of agents

sists in booking the vacation, needs to be addressed by some decisions, e.g. on some alternative transport services and on some alternative accommodations. The main goal (g_0) is split into independent sub-goals: the vacation must be cheap (g_1), the destination must be attractive (g_2), and the quality of service must be high (g_3). This sub-goal is reduced to further sub-goals: the quality of service depends on the quality of accommodation (g_4) and the quality of transport (g_5). While the high-level goals are *abstract*, i.e. they just reflect the user's needs, the low-level goals are *concrete*, as they provide criteria for evaluating different alternatives. The information held by the agent about location may be expressed by means of predicates such as: `Public`(x) (accommodation x is accessible by public transports), and `Safe`(x) (transport x is safe).

Figure 2 provides a simple graphical representation, called *influence diagram*, of the decision problem for this example. The elements of the decision problem *values* (represented by rectangles with rounded corners), *decisions* (represented by squares) and *knowledge* (represented by ovals) are connected by arcs where predecessors are independent and affect successors. The problem is a multi-attribute decision problem captured by a hierarchy of values where the abstract values (represented by rectangles with rounded corner and double line) aggregate the values in the lower levels.

The influence diagram displays the structure of the decision problem, and is kept within the IKB of the agent in our model (see next Section). In addition, the GUI allows the user to communicate user-specific details of the decision making, in particular preferences (e.g. Paris is the most preferred destination) and some

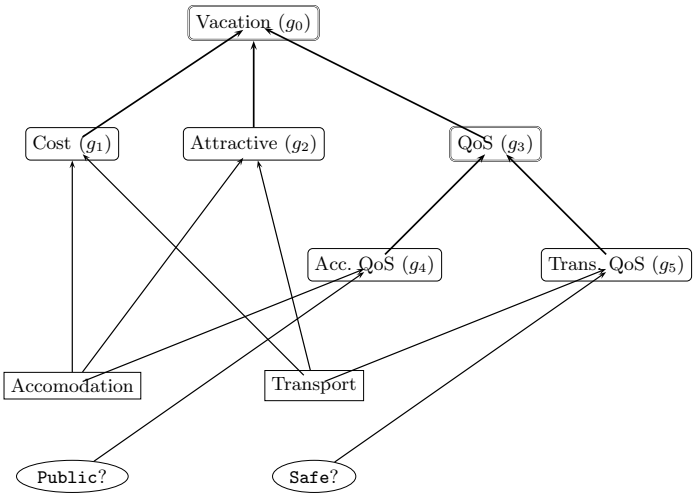

Fig. 2. Influence diagram to structure the decision

constraints (e.g. the user's budget). These preferences are also recorded in the agent's IKB and reasoned upon by the IDMM.

In order to support the matching of service descriptions and concrete services offered by providers, it is obvious that the services offered by the providers have to carry sufficient descriptive information to support automated search and composition. For this purpose, the Web Service Modeling Ontology (WSMO) [9] provides a conceptual framework and a formal language for semantically describing all relevant aspects of (Web) services in order to facilitate the automation of discovering, combining and invoking electronic services, in particular over the Web.

In our example, the "Vacation" ontology, defining a trip and underlying concepts, can be represented in WSML, which is a language to describe ontologies and semantics for Web Services [10]. The definition of the ontology is based on the "International Train Ticket" ontology from the WSMO Use Case "Virtual Travel Agency" [11]. Our ontology defines vacation by plane. It reuses the flight concept and adapts it to define the accommodation concept and some different sub-concepts such as hotel and IYH (International Young Hosteling). Figure 3 represent a WSMO goal, i.e. an abstract representation of the type of service requested by the user.

In the remainder of the paper, we will focus on the viewpoint of the agent requesting/searching for the composite service.

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

namespace {_"http://www.argugrid.eu/travel/goal#",
  dO _"http://www.argugrid.eu/travel/domainOntology#",
dc _"http://purl.org/dc/elements/1.1#"}

goal _"http://www.argugrid.eu/travel/goal.wsml"
nfp
dc#title hasValue "Goal"
dc#contributor hasValue "WG2"
                dc#description hasValue
                "Express the goal of buying a plane ticket"

endnfp
importsOntology _"http://www.argugrid.eu/travel/domainOntology.wsml"
capability goalCapability
postcondition
definedBy
?ticket[
dO#from hasValue ?from,
dO#to hasValue ?to,
dO#cost hasValue ?cost
] memberOf dO#Ticket.
```

Fig. 3. Domain Ontology “Vacation” namespace

4 Individual Decision Making

The first module is the *Individual Decision Making Module* (IDMM), supporting the reasoning about the kind of services which can be requested. The IDMM is supported by the concrete data structures in the *Individual Knowledge Base* (IKB). Decisions are made according to the user's requirements about the services, the alternative types of services, and the users' preferences and constraints.

In ARGUGRID, the IDMM is built using a concrete argumentation framework for practical reasoning [12], which in turn adopts the dialectical, argumentation tools of [13, 14]. The implementation of this argumentation framework, called MARGO[6], uses the implementation of [13, 14] in the CaSAPI system [15].

In MARGO, a logic-based language is used as a concrete data structure for holding the knowledge, goals, and available decisions of the agent. *Priorities* are attached to these items corresponding to the probability of the knowledge, the preferences between goals, and the expected utilities of decisions. These concrete data structures consist of information providing the backbone of arguments. Due to the abductive nature of decision making, arguments are built by reasoning backwards. To be intelligible, arguments are defined as tree-like structures. Since an ultimate choice amongst various admissible sets of alternatives is not always possible, we have adopted a "credulous" semantics [13].

The user also provides, through the GUI, users' preferences and constraints. For example, the user may specify that the cost (g_1) is more important to the user than the quality of service (g_3) as far as the vacation (g_0) is concerned. *Priorities* are attached to goals, decisions and knowledge in an influence diagram to represent the preferences over goals, the expected utilities of decisions, and the uncertainty of the knowledge.

Influence diagrams and priorities are recorded within the agent's IKB and reasoned upon by the IDMM. The concrete data structures in the IKB contain the abstract representation of service(s), i.e. some single (or a workflow of) abstract or partially instantiated services. The selection of abstract services is made according to the knowledge, the goals of the user, the alternative types of services, and the priority over them. For instance, the IKB corresponding to the influence diagram of Figure 2 will contain the three following rules:

$r_{0123} : g_0 \leftarrow g_1, g_2, g_3$

$r_{012} : g_0 \leftarrow g_1, g_2$

$r_{023} : g_0 \leftarrow g_2, g_3$

expressing that achieving g_1, g_2, and g_3 is ideally required to reach g_0 (cf r_{0123}), but this can be relaxed: according to r_{012} (resp. r_{023}), achieving the goal g_1 (resp. g_3) and the goal g_2 is enough to reach g_0. Priorities amongst goals, namely that g_1 matters more than g_3 to the user, is represented in MARGO by means of priorities over rules, namely r_{0123} has priority over r_{012} which has priority r_{023}. The priority of such rules corresponds to the relative importance of the combination of (sub)goals in the body as far as reaching the goal in the head is concerned.

[6] http://margo.sourceforge.net

The IDMM interact with the GUI, through the CM, by being informed that a workflow must be set up, and by presenting this workflow which is (or is not) set up. The IDMM interact with the SDMM by asking the instantiation of the abstract or partially instantiated workflow, and by being informed when a concrete workflow is (or is not) set up.

In this way, the IDMM module shifts from the goals, the preferences, and the constraints provided by the user to an abstract representation of services in a workflow (or a single service, as appropriate). For instance,
$\langle g_0, [\text{Transport}(x)], [\text{Safe}(x)]\rangle$ is an argument concluding that the goal related to the vacation is reached if we suppose that the transport is safe. This is then turned into a concrete representation (choice of x fulfilling the constraint) by the SDMM.

5 Social Decision Making

The second module is the *Social Decision Making Module* (SDMM), reasoning about the concrete instances of services which can be requested. Decisions are made according to user's requirements about the service providers, the alternative concrete services, and preferences over them. For this purpose The SDMM is supported by the concrete data structures in the *Social Knowledge Base* (SKB) and in the *Commitments*.

The commitments are internal data structures which contain propositional and action commitments involving the agent, namely with the agent being either the debtor or the creditor. These data structures are shared by the SIM and the SDMM. Concretely, the commitments may contain the concrete representation of atomic or composite services and the representation of the partners exchanged during the dialogues, while the SKB contains the concrete representation of atomic or composite services provided by the agent. Moreover, the SKB contains preferences about the services and the partners. The selection (resp. the suggestion) of concrete services is made according to the user's requirements (resp. competencies) about the alternative concrete services, the information about the partners, and preferences over them. In ARGUGRID, the SDMM, like the IDMM, is built upon MARGO.

The SDMM interacts with the IDMM by being informed that an abstract workflow must be instantiated, and by informing it when a concrete workflow is (or is not) set up. The SDMM interacts with the SIM, by notifying it that the agent needs to play a certain role in an interaction, by being informed by the SIM when some offers/proposal must be evaluated, by informing the SIM when the offers/proposals are evaluated, and finally by being informed by the SIM of the final result of an interaction.

In this way, the SDMM reasons and takes decision about the concrete instances of services which can be provided.For instance: $\langle g_4, [\text{Transport}(c)], [\text{Safe}(c)]\rangle$ is an argument concluding that the goal related to the quality of the transport service is reached if we suppose that the concrete service c is safe.

6 Social Interaction

The third module, the *Social Interaction Module* (SIM), drives the communication and interactions by the adherence to protocols. These protocol are concrete data structures and are stored in the *Protocol Library* (PL).

In ARGUGRID, the SIM uses the Lightweight Coordination Calculus (LCC), a logic-based language allowing to drive all social interactions [16]. For this purpose, the concrete data structures in the Protocol Library (PL) contain the LCC representation of the interaction protocols. Decisions required to conduct the interaction are provided by the SDMM. For those readers unfamiliar with LCC, it may be helpful to note that the LCC is a declarative logic programming language in the style of Prolog, augmented with CCS (a process calculus for communicating systems). The usefulness of interaction protocols as first-class computational entities is discussed in [17]. Though the use of dynamic interactions protocols is possible, we restrict ourselves to static ones.

$\mathcal{P}$	$\in$ Protocol	$::= \langle S, A^{\{n\}}, K \rangle$
$\mathcal{A}$	$\in$ AgentClause	$::= \theta :: \text{op}.$
θ	$\in$ AgentDefinition	agent(R,Id)
op	$\in$ Operation	no op
		$\text{—}\theta$
	(Precedence)	—(op)
	(Send)	$\text{—}M \Rightarrow \theta$
	(Receive)	$\text{—}M \Leftarrow \theta$
	(Sequence)	—op1 then op2
	(Parallelization)	—op1 par op2
	(Choice)	—op1 or op2
	(Prerequisite)	$\text{—}(M \Rightarrow \theta) \leftarrow \psi$
	(Consequence)	$\text{—}\psi \leftarrow (M \Rightarrow \theta)$
M	$\in$ message	$::= \langle m, \mathcal{P} \rangle$

Fig. 4. An Abstract Language of the Protocol Language

Figure 4 defines the syntax of the LCC protocol language. A protocol consists of a set of agent clauses, $A^{\{n\}}$. An agent clause is the series of communicative actions expected to be performed by an agent adopting the role defined by the agent definition. This agent definition consists of a role (R) and unique identifier (Id). The agent definition is expanded by a number of operations. Operations can be classified in three ways: actions, control flow, and conditionals. Actions are the sending or receiving of messages, a no op (i.e no action is taken), or the adoption of a role. Control flow operations temporally order the individual actions. Actions can be sequentially ordered, performed simultaneously without regard to order, or given a choice point. The definition of the double arrows denote messages M being sent and received. On the left-hand side of the double arrow is the message and on the right-hand side is the other agent involved in the interaction.

Constraints can fortify or clarify the semantics of the protocols. Those occurring on the left of the $\leftarrow$ are post-conditions and those occurring on the right are preconditions. The symbol ψ represents a first order proposition. For example, an agent receiving a protocol with the constraint to believe a proposition `s` upon being informed of `s` can infer that the agent sending the protocol has a particular semantic interpretation of the act of informing other agents of propositions. The operation $(M \Rightarrow \theta) \leftarrow \psi$ is understood to mean that message M is being sent to the agent defined as θ on the condition that ψ is satisfiable. The operation $\psi \leftarrow (M \Rightarrow \theta)$ means that once M is received from agent, ψ holds.

Figure 5 represents the set of clauses for the requesting agent in the well-known Contract Net Protocol. The clause for the provider role is the complement of the requestor's. The role being used is `requester(Providers,X)`, `Providers` being a list of potential providers of the service `X`. A cfp is sent to the first provider in the list and the requester waits for a response. When he gets one, the role recurses on the tail of the list of providers, `a(requester(T,X),Req)`. The actions of the requestor are split into two clauses to enable control over the recursion over the list of potential providers. Hence, when the list is exhausted, he moves to the next role `a(requester1(Providers1,X),Req)`. The constraint `interestedparties(Providers1)` returns the list of providers who tendered proposals. This is the responsibility of the SDMM. The second clause `a(requester1(Providers1,X),Req)` now handles the next stage of the contract net protocol, i.e. the rejection and acceptance of the proposals given. Similar to the last clause, this is done by recursing over the list of Providers, `Providers1`. After this, we go to the final stage defined by the clause `a(requester2(WinningProvider,X),Req)`. This clause defines the requesters behaviour as waiting for the provider to send the results of the provision of the service.

In this way, the SIM manages the sequence of messages defined by the protocol.

7 Scenario

Figure 6 represents the sequence diagram associated with the service requestor in the example outlined in Section 3. The IDMM is informed by the GUI that a workflow must be set up. The IDMM builds the abstract workflow and asks the instantiation of this abstract workflow to the SDMM. The SDMM receives the abstract workflow from the IDMM and calls the SIM for it to enable the agent to play the role of requestor in an interaction. In order to play this role in a Contract Net Protocol, the SIM informs (resp. is informed by) the CM when some messages must be sent (resp. are received). The SDMM is informed by the SIM when some offers must be evaluated, evaluates the offers and selects the best one, acceptable wrt the constraints. Being informed by the SDMM when the evaluation is done, the SIM updates the commitments stores, drives the interaction and informs the SDMM of the final result.

a(requester(Providers,X),Req) ::=
(cfp(X) ⇒ a(provider(X),Prov) ← Providers = (Prov—T) then
(refuse(X) ⇐ a(provider(X),Prov)
or assert(propose(Prov,X)) ← propose(X) ⇒ a(provider(X),Prov))
then a(requester(T,X),Req))
or a(requester1(Providers1,X),Req) ← interestedparties(Providers1).

a(requester2(WinningProvider,X),Req) ::= failure ⇐ a(provider(X),Prov)
or inform-result(X) ⇐ a(provider(X),Prov)
or inform-done ⇐ a(provider(X),Prov).

a(provider(X),Prov) ::= cfp(X) ← a(Y,Req) then (refuse(X) ⇒ a(Y,Req)
or (propose(X) ⇒ a(Y,Req) then
reject-proposal(X) ⇐ (requester1(Providers1,X),Req)
or (accept-proposal(X) ⇐ a(requester1(Providers1,X),Req)
then failure ⇒ a(requester1(Providers2,X),Req)
or inform-result(X) ⇐ a(requester1(Providers2,X),Req)
or inform-done ⇐ a(requester1(Providers2,X),Req)))).

Fig. 5. Representation of the ContractNet Protocol

In the concrete scenario depicted here, the IDMM is informed by the SDMM that the workflow cannot be instantiated since the IYH in Paris is fully booked. The IDMM modifies the abstract workflow to choose any cheap hostel and asks the instantiation of this new workflow to the SDMM. At the end of this process, the IDMM is informed that the workflow is instantiated and forwards the outcome to the GUI. Since the sequence diagram associated with a service provider is similar, we will not detail it.

8 Related work

A plethora of models and architectures for building agents are already available [18, 19]. This renders the task of reviewing all the related work in this area gigantic. As a result, to make our task more manageable, in this section we identify only a subset of existing models that we believe to be most relevant to compare with our agent model and architecture.

The Belief-Desire-Intention (BDI) model of agency is the best known model of agents [20]. However, the simplifying assumptions made to implement modal logic specifications of BDI agents meant that they lack of a strong theoretical underpinning [21].

The KGP model [22] adopts Knowledge, Goals, and Plans as the main component of an agent state. Contrary to the BDI model, there is no gap between the logical specification of KGP agents and their implementations. For this purpose, this model uses computational logic frameworks, extending, in many cases, logic programming with dynamic priorities [23]. However, it deals only partially with priorities as required by service composition applications, namely only with

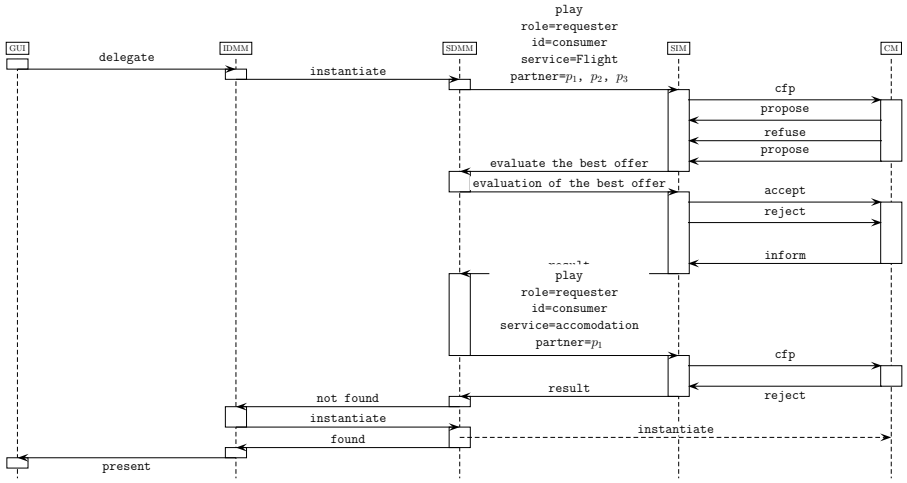

Fig. 6. Sequence diagram of the service requestor

preferences amongst goals, but not with probability of knowledge and expected utilities of alternative services. For this purpose, we have provided here a suitable revised representation of knowledge, goals and actions built upon the approach of [13, 14].

An important characteristic of our architecture is that it provides a set of tools supporting the individual reasoning, the social reasoning, and the management of the interactions with other agents for free. Many FIPA-compliant platforms, for example FIPA-OS [24], or ZEUS [25], do not commit to a model of agency but support only the interaction of the agent with the environment. In such platforms the programmer has to develop from scratch the reasoning capabilities of the agent. Other FIPA compliant approaches, such as JADE [26] and 3APL [27], make use of tools that support the reasoning capabilities of the agent. Unlike our logic-based approach which is closer to the specification of an agent, JADE can be used with Jadex [28] to implement the BDI model using a Java API. On the other hand, 3APL uses a logic-based language which, like BDI, is linked to a modal-logic framework [27].

An approach similar to our is the recent development of the Jason [29] platform that implements AgentSpeak(L) agents. Like Jason agents, our agents attempt to narrow the gap between the specification and executable model of an agent. Also like Jason agents, agent interactions can be verifiable because our model is based on a formal computational framework for individual reasoning, social reasoning and social interactions. However, our agents differ from Jason agents in that AgentSpeak(L) is based on the BDI model.

9 Conclusions and future work

In this paper, we have adopted an argumentative model of agents able to select and compose services. For this purpose, we have proposed a modular agent architecture using three main modules dedicated respectively to decision making (the individual decision making module), negotiation (the social decision making module), and communication (the social interaction module). We have outlined how this architecture is instantiated in the ARGUGRID project, by means of the argumentative decision making tool MARGO (for the first two modules) and by the LCC tool for enforcing protocol conformance (for the last module). We have illustrated our architecture and its operation in the context of a virtual travel agency example.

We have only presented one model of the agent's mind. However the framework is designed to allow agents with different mental models and created by third-parties to participate in the system. However, even in such an open system, our assumption that the agents share ontologies and dialogical framework (i.e interaction protocols) remains.

Future investigations will consider argumentative protocols for rendering the negotiation capabilities of the agents more powerful. Moreover, we are currently developing a model of internal dialectic between the individual decision making module and the social decision making module. In our example, the instantiation of the first workflow fails, since the IYH is fully booked. This information could be useful for the individual decision making module to provide a new abstract workflow. Additionally, we are currently researching the ability of agents to negotiate new interaction protocols and issues associated with trust and reputation.

Acknowledgements

This work is supported by the Sixth Framework IST programme of the EC, under the 035200 ARGUGRID project.

References

1. Morge, M., McGinnis, J., Bromuri, S., Toni, F., Mancarella, P., Stathis, K.: Vers une architecture modulaire d'agent argumentatif pour la composition de services. In: in Proc. of the of 15th Journées Francophones sur les Systèmes Multi-Agents (JFSMA). (2007) 10 pages Á paraitre.
2. Singh, M.P., Huhns, M.N.: Service-Oriented Computing. Semantics, Processes, Agents. Jonh Wiley & Sons, Ltd (2005)
3. Prakken, H., Vreeswijk, G.: logical systems for defeasible argumentation. In: Handbook of Philosophical Logic. Volume 4. Kluwer Academic Publishers (2002) 219–318
4. Morge, M., Mancarella, P.: The hedgehog and the fox. An argumentation-based decision support system. In: Proc. of the Quatrième Journées Francophones Modèles Formels de l'Interaction. (2007) 357–364

5. Morge, M.: Systme dialectique au travers duquel les agents jouent et arbitrent. vers une prise de dcision collective et dbattue. In: Actes des Journes Francophones sur les Systmes Multi-Agents, Herms (2005) 115–127
6. Amgoud, L., Dimopoulos, Y., Moraitis, P.: A unified and general framework for argumentation-based negotiation. In: Proc. 6th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'07), Honolulu, Hawaii (2007)
7. Bracciali, A., Demetriou, N., Endriss, U., Kakas, A., Lu, W., Stathis, K.: Crafting the mind of PROSOCS agents. Applied Artificial Intelligence **20**(2–4) (2006) 105–131
8. Fornara, N., Colombetti, M.: Operational specification of a commitment-based agent communication language. In: Proc. of the 1st international joint conference on autonomous agent and multi-agent systems, ACM Press (2002) 535–542
9. Lausen, H., Polleres, A., Roman, D.: Web service modeling ontology (WSMO). Technical report, W3C (2005) http://www.w3.org/Submission/WSMO/.
10. de Bruijn, J., Fensel, D., Keller, U., Kifer, M., Lausen, H., Krummenacher, R., Polleres, A., Predoiu, L.: Web service modeling language (wsml). Technical report, W3C (2005) http://www.w3.org/Submission/WSML/.
11. Stollberg, M., Lara, R.: D3.3 v0.1 WSMO use case "virtual travel agency". Technical report, WSMO (2004) http://www.wsmo.org/2004/d3/d3.3/v0.1/.
12. Morge, M., Mancarella, P.: The hedgehog and the fox. An argumentation-based decision support system. In: Proc. of the Fourth International Workshop on Argumentation in Multi-Agent Systems. (2007) 55–68
13. Dung, P.M., Kowalski, R.A., Toni, F.: Dialectic proof procedures for assumption-based, admissible argumentation. Artificial Intelligence **170**(2) (2006) 114–159
14. Dung, P.M., Mancarella, P., Toni, F.: A dialectic procedure for sceptical, assumption-based argumentation,. In: Proc. of the 1st International Conference on Computational Models of Argument (COMMA 2006), IOS Press (2006)
15. Gartner, D., Toni, F.: CaSAPI: a system for credulous and sceptical argumentation. In Simari, G., Torroni, P., eds.: Proc. Workshop on Argumentation for Non-monotonic Reasoning. (2007) 80–95
16. Robertson, D.: Multi-agent coordination as distributed logic programming. In Springer-Verlag, ed.: Proc. of the 20th International Conference on Logic Programming (ICLP), Saint-Malo, France (2004) 416–430
17. Miller, T., McGinnis, J.: Amongst first-class protocols. In: in Proc. of ESAW 2007, Athens, Greece (October 2007)
18. Bordini, R.H., Dastani, M., Dix, J., Seghrouchni, A.E.F., eds.: Multi-Agent Programming. Kluwer Academic Publishers (May 2006)
19. Ricordel, P.M., Demazeau, Y.: From analysis to deployment: A multi-agent platform survey. In: Engineering Societies in the Agents World: First International Workshop, ESAW 2000. Volume 1972/2000 of Lecture Notes in Computer Science., Berlin, Germay, Springer Berlin / Heidelberg (February 2000) 93–105
20. Rao, A.S., Georgeff, M.P.: Modeling rational agents within a BDI-architecture. In Allen, J., Fikes, R., Sandewall, E., eds.: Proc. of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR), Morgan Kaufmann publishers Inc.: San Mateo, CA, USA (1971) 473–484
21. Rao, A.S.: Agentspeak (l): BDI agents speak out in a logical computable language. In Hoe, R.V., ed.: in Agents Breaking Away, 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96). Number 1038 in Lecture Notes of Computer Science, Springer-Verlag (1996) 42–55

22. Kakas, A.C., Mancarella, P., Sadri, F., Stathis, K., Toni, F.: The KGP model of agency. In: Proc. of ECAI. (2004) 33–37
23. Prakken, H., Sator, G.: Argument-based logic programming with defeasible priorities. Journal of Applied Non-classical Logics **7** (1997) 25–75
24. Poslad, S., Buckle, P., Hadingham, R.: The fipa-os agent platform: Open source for open standards. In: Proc. of PAAM'00. (2000) 255–368
25. s. Nwana, H., Ndumu, D.T., Lee, L.: Zeus: An advanced tool-kit for engineering distributed multi-agent systems. In: Proc. of the Practical Application of Intelligent Agents and Multi-Agent Systems, London (1998) 377–392
26. Fabio Luigi Bellifemine, Giovanni Caire, D.G.: Developing Multi-Agent Systems with JADE. Wiley (February 2007)
27. Dastani, M., van Riemsdijk, B., Dignum, F., Meyer, J.J.: A programming language for cognitive agents: Goal directed 3APL. In: Proc. of the First Workshop on Programming Multiagent Systems: Languages, frameworks, techniques, and tools (ProMAS03). (July 2003)
28. Pokahr, A., Braubach, L., Lamersdorf, W.: Jadex: Implementing a BDI-infrastructure for JADE agents. EXP - in search of innovation (Special Issue on JADE) **3**(3) (9 2003) 76–85
29. Rafael H. Bordini, Jomi Fred Hubner, M.W.: Programming Multi-Agent Systems in AgentSpeak using Jason. Wiley (August 2007)