DOI 10.12694/scpe.v15i1.966
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 15, Number 1, pp. 65–78. http://www.scpe.org

A DISTRIBUTED AGENT-BASED DECISION SUPPORT FOR CLOUD BROKERING

ALBA AMATO* AND SALVATORE VENTICINQUE†

Abstract. In goal oriented problems, decision-making is a crucial aspect aiming at enhancing the ability to make decisions by full autonomous, or supervised software agents. Agents usually resolve a huge number of evaluations and decisions without the user intervention. Just the remaining uncertainty should be left to the user's attention. In this paper we present a complete constrained multi-objectives optimization problem, modeled as an agent based decision support systems, that helps to choose the Cloud proposals that best satisfy the needs of the user, among the ones offered by known vendors. In particular we focus on a distributed solution that exploits the multi-agents programming model and the Cloud elasticity to comply with computational requirements of delivering such brokering at Cloud service level.

Key words: Multi-Agents systems, Cloud provisioning, Brokering, Multi-Criteria Decision

78

1. Introduction. The research of solutions for delegation to intelligent software agents in information systems has implied the need to develop systems able to act effectively: the systems must be able to act independently and follow the choice that best represents their interest during the interaction with a human being or another system.

To successfully interact, is essential for agent the ability to cooperate, coordinate, and negotiate with each other. In particular the research for even more intelligent systems implies that the complexity of tasks delegating to computers has also grown steadily leading to the need to develop systems able to act effectively and independently and to make decisions. In fact delegation and intelligence imply the need to build computer systems that can act effectively in a way that represents the best interests of the user while interacting with other humans or systems [23].

In this kind of goal oriented problem, decision-making is a crucial aspect aiming at enhancing the user's ability to make decisions. The decision has to be made among possible alternatives taking into account all the certainty and uncertainty considerations and using a prescribed set of decision criteria.

The reason for this intensive interest in decision-making [35] is that the metaphor of autonomous problem solving entities cooperating and coordinating in order to achieve their desired objectives is an intuitive and natural way of the problem solving. Moreover, the conceptual apparatus of this technology provides the powerful and useful set of computational structures and processes for designing and building the complex software applications.

The multi-agent techniques combined with decision-making tools can help decision makers to deal with the problems of the information overload. In addition, it is possible to make better and wider use of the existing knowledge by extracting value from large volumes of data. An additional challenge of the research is the possibility for an intelligent agent of being "rational" by having preferences about the goals to be achieved on the basis of interests expressed by the user. In particular the application of agent-based decision support helps users to make the right choice improving their satisfaction.

Enhancing software agents with decision making features and addressing them like rational entities is a topic of keen interest in designing agents and, which has been researched intensively [16] , [32].

The brokering problem, described in [1, 5] is a decision problem that consists of choosing the best proposals among the number of offers, which have been received from different providers, who answered to the same call [4]. The brokering has been addressed as a complete multi-objective constrained problem. The solution of this problem is one of the main objective of mOSAIC project [21].

To solve the brokering problem, we designed and developed Cloud Agency, a Multi-Agent System (MAS) that has the main task of dynamically selecting a set of Cloud resources, from different vendors, that best fits users' requirements. It also allows for vendor agnostic management and monitoring of Cloud infrastructures, where legacy or mOSAIC application are deployed as presented in [11]. It is compliant with the NIST definition

*Department of Industrial and Information Engineering, Second University of Naples, Aversa, Italy (alba.amato@unina2.it).
†Department of Industrial and Information Engineering, Second University of Naples, Aversa, Italy(salvatore.venticinque@unina2.it).

of cloud broker as defined in [20], that is an entity that manages the use, performance and delivery of cloud services, and negotiates relationships between Cloud Providers and Cloud Consumers. Cloud Agency actively performs cloud broker functions at the platform level.

In [3] we demonstrated the feasibility of the specific approach in the case of a single user, considering the number of existing providers in the current Cloud market, in the case that all the providers have at least one proposal. However the centralized approach is not effective in the case of brokering delivered as a service to a huge number of users. Performance bottleneck was observed so we investigated a new solution to support either a greater number of decisions and the dimension of the single decision problem.

To provide the brokering facility at Application as a Service level we have to overcome the performance limitations of the mOSAIC Cloud Agency solution, starting from the identification of new requirements. First of all it needs to identify the issues introduced by the new scenario, but also to take in consideration the available technologies for designing and implementing a new engineered solution.

About the issues, we need to take into account the number of service users who can access contemporary the service by multiple requests. We have to consider that in a service oriented context, not only human users, but also applications and robots will be able to invoke the service producing different kind of workloads. The new workload can vary in dimension and it can also dynamically change during the day, with regular or unforeseeable bursts on special periods. For this reason we have to share the workload over a distributed computing infrastructure, and we need to grant that both the infrastructure and the application will scale dynamically.

In this paper we present a scalable distributed multi-users version of the brokering service provided by Cloud Agency. Such new Broker As A Service solution exploits the capability of a distributed environment and addresses related issues. The idea is to divide the brokering problem into simple tasks, which are distributed to independent and collaborative agents, scaling dynamically in number, together the computing infrastructure, to support unforeseeable workloads produced by the interactions with large groups of users. In addition we evaluate and discuss performance results and effectiveness of a first prototype implementation.

2. Related Work.

2.1. Decision-making. According to [34] the Multi-Criteria Decision Making (MCDM) is divided into Multi-Objective Decision Making (or MODM) and Multi-Attribute Decision Making (or MADM). As described in [28] MODM studies decision problems in which the decision space is continuous. The goal is to find the best alternative among endless alternatives known implicitly.

On the other hand, MADM concentrates on problems with discrete decision spaces. In these problems the set of decision alternatives has been predetermined. Paper [9] claims that, although MADM methods may be widely different, many of them have certain aspects in common, such as the notions of alternatives, and attributes (or criteria, goals). In fact the general approach consists of the utilization of information about the problem (factual elements) together with the preferences express by the decision maker (value elements) to find the best compromise that will help to select the most consistent alternative, according to his preferences.

A decision making process typically has five steps that are summarized as follow [7]: Identify the problem or opportunity, Develop alternative, Evaluate alternative, Choose and implement the best alternative, Evaluate the decision.

In general, the outcome of a decision (action) depends on the context in which the decision is implemented, but a very general decision rule applied to MCDM problems consists in the identification of all efficient alternatives and the selection of the solution using the information of preference that the decision maker makes available. According to [6] the "optimum" solution corresponds to the feasible point for which the objective function achieves an optimum value.

The current role of multi-objective optimization in various sectors is becoming increasingly relevant. As with most problems, the objectives to be considered are many and often contradict each other. In particular, our approach is a MultiCriteria Decision Analysis/Aiding (MCDA) as addresses mainly discrete problems with not very large (combinatorial) sets of alternatives. For combinatorial problems find rules for decision making. Constrains are taken into account implicitly: into set of criteria and/or alternatives.

The multi-criteria decision-support is a valuable methodological tool in the processes of decision-making support. The multi-criteria decision-support is intended to provide to decision-makers some tools, which help

them in resolution of a problem of choice among alternatives, when numerous and often contradictory points of view must be taken simultaneously in consideration.

In general there is no decision (solution, action) that is the best one compared to all involved points of view. Besides there are situations where, as many factors have to be considered, the available alternatives are too much different or even not comparable. In addition, we are interested in those contexts where there are different stakeholders that give a different value to the several factors to be considered, that represent the preferences for criteria. Hence the problem is how to measure the importance of these criteria and how to combine these values.

In this scenario, one of the main implications of multi-criteria decision support has been the loss of optimality. In fact, given the presence of heterogeneous objectives often in conflict with each other, a cost benefit analysis is not feasible. Usually it is not possible to find solutions that simultaneously pursue all the objectives. The decision problem is solved looking for the most satisfactory solution, or "more consistent" with the logic of decision makers. The choice must therefore be performed among those solutions that realize a certain level of achievement of the various objectives, such that it is not possible to improve one of them without degrading some others.

According to [22], multiple criteria modeling methodology can address the following set of issues :

- Issue 1: Object of the decision, including the definition of the set of potential actions A and the determination of a problematic applied on A.
- Issue 2: Modeling of a consistent family of criteria assuming that these criteria are non-decreasing value functions, exhaustive and non-redundant.
- Issue 3: Development of a global preference model, to aggregate the marginal preferences on the criteria.
- Issue 4: Decision-aid or decision support, based on the results of Issue 3 and the Issue 1.

In Issue 1, Roy [22] distinguishes four referential problematics, each of which does not necessarily preclude the others. These problematics can be addressed separately, or in a complementary way in all phases of the decision-making process. They are listed below:

a: Choosing one action from A (*choice*).
b: Sorting the actions in well defined categories which are given in a preference order (*sorting*).
c: Ranking the actions from the best one to the worst one (*ranking*).
d: Describing the actions in terms of their performances on the criteria (*description*).

2.2. The Brokering Problem. There were some efforts aimed on solving the brokering problem. In [8] an architecture is presented for a federated Cloud computing environment named InterCloud to support the scaling of applications across multiple Cloud providers using a Cloud Broker for mediating between service consumers and Cloud coordinators for an allocation of resources that meets QoS needs of users.

Sim [24] proposes an extension of the alternate offers protocol that supports multiple complex negotiation activities in interrelated markets between user agents and broker agents, and between broker agents and provider agents.

In [19] an architectural design of a framework capable of powering the brokerage based Cloud services is presented. It is currently being developed in the scope of OPTIMIS, an EU FP7 project. The cited paper introduces the problem and the architectural design but it does not provide an implementation or algorithms to achieve the brokering.

Our brokering solution, overcomes the problems of the centralized broker so allowing a multi-user utilization. Besides it allows the concurrency, is highly scalable, open and available to enable easy extensions of existing components and addition of new components.

2.3. BDI agents. In BDI architecture [15],Beliefs, Desires, Intentions are the basic components of an agent that should be able to operate in a dynamic, uncertain world. Beliefs represent agent's knowledge of the world, Desires (or goals) represent what the agent wants and Intentions are a set of plans used to describe how an agent achieve his goals. BDI agents are adaptive in the sense that they can quickly reason and react to asynchronous events acting accordingly to them.

According to Wooldridge [32], the process of practical reasoning in a BDI agent is summarized in Figure 3.1 As this figure illustrates, there are seven main components to model a BDI agent:

- a set of current beliefs, representing information the agent has about its current environment;
- belief revision function, (brf), which takes a perceptual input and the agent's current beliefs, and on the basis of these, determines a new set of beliefs;
- an option generation function, (options), which determines the options available to the agent (its desires), on the basis of its current beliefs about its environment and its current intentions;
- a set of current options, representing possible courses of actions available to the agent;
- a filter function (decision), which represents the agent's deliberation process, and which determines the agent's intentions on the basis of its current beliefs, desires, and intentions;
- a set of current intentions, representing the agent's current focus (those states of affairs that it has committed to trying to bring about);
- an action selection function (execute), which determines an action to perform on the basis of current intentions.

BDI agents often have to make decisions, about which plan is used to achieve a goal, and in which order goals are to be achieved. Besides The BDI approach has proved valuable for the design of agents that operate in dynamic environments. It offers a higher level of abstraction by explicitly allowing beliefs to have a direct impact upon the agents behavior. This means the agents can respond flexibly to changing circumstances despite incomplete information about the state of the world and the agents in it [12].

2.4. Mult-agent decision support systems. Intelligent agents, when combined with Decision Support System, provide powerful support in solving difficult applied problems that are often real-time, involve large amounts of distributed data, and benefit from complex reasoning.

In fact it is evident [10] that a methodology for solving multi-criteria decision support can be applied to the agents modeling problems [13]: a consistent family of criteria (Issue 2) can sufficiently model the world with respect to the object of the decision (Issue 1). A global preference model (Issue 3) is able to guide agents' decisions. Finally, agents' actions will be realized based upon the decision support step (Issue 4).

Multiple criteria methodologies could contribute as a methodological background for agents modeling as well as a tool for their real-time implementation [29]. However, in this work we stress their potential for modeling support. In this respect, the potential contributions lie across two levels:

1. Agent design level: The multiple criteria modeling can respond to the representation issue discussed earlier. Existing methods can provide agents with critical decision making elements.
2. Multi agent level: Existing methods can profile decision makers (DMs), thus facilitate coordination – negotiation. In addition, DM's profiles can be exploited to assign roles or tasks to agents.

3. Problem Formulation. Problem formulation represents the first step of the decision process. The way of approaching the problems means defining a set of actions in order to act in an organized and methodical manner in the situations where a problem appears. In that way, the plan defines problem solving procedures according to the typology and the level of complexity of the problem itself. In particular in our approach the problem is modeled as shown in Figure 3.1. Some elements of the agents model will be introduced here and detailed in the next sections.

Using a simple definition, an agent can be seen as an entity that perceives the environment and acts rationally trying to achieve goals after that input perceptions and a knowledge base are given.

3.1. Environment. An agent is anything that can be viewed as perceiving environment through sensors and acting upon that environment through effectors. Intelligence of agents is the ability to adapt to the environment. In other words, as the environment changes an intelligent agent is able to adapt its behaviour to the new environment [25].

Environment represents information source to which react. This environment might be a real physical environment, a virtual one (so only existing in a computer), or a mix of both. Environment can be changed by agents through actions.

Arguably, the capability to adapt to the environment, means that just about any agent is an intelligent agent. Given an environment that can be in two possible states, an agent that is able to adapt its behaviour to both states is intelligent. An intelligent agent is then an agent that is able to adapt to a large number of states of its environment.

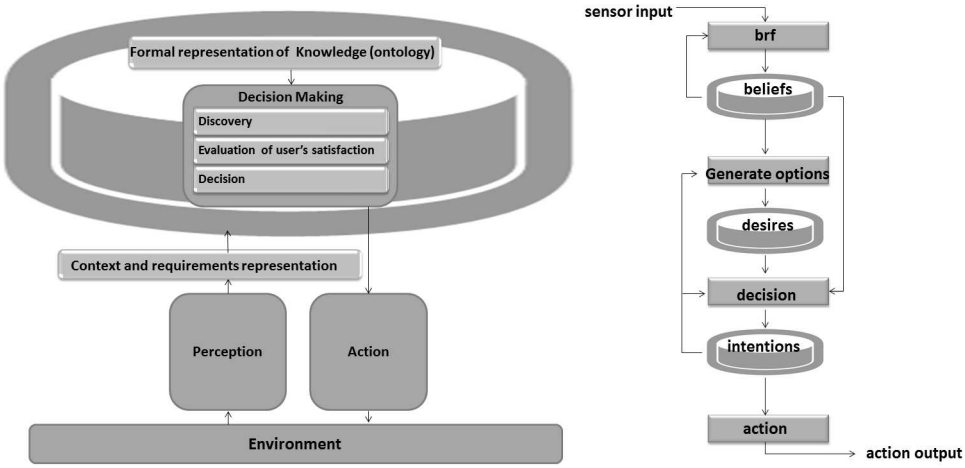

FIG. 3.1. *Decision-making support methodology*

In the brokering problem the environment is represented by the different proposals that agents perceive.

3.2. Knowledge representation. Knowledge is a set of representations of facts of the world and constitutes the base for reasoning. In facts intelligent agents need knowledge about the world for making good decisions.

Knowledge also includes the awareness and control of environment, but obviously an agent will not have complete control over its environment. In fact we always assume an agent has an imperfect view of its environment, and hence it cannot be certain if its actions always have the intended result. It will have at best partial control, in that it can influence it. The agent takes sensory input from the environment, and produces as output actions that will affect it.

Perceptions, state and actions of an agent constitute the agent's knowledge. If an agents is able to choose action sequences that maximize changes of the environment according to its own expected preferences, it can be defined a rational agent. Rationality is an important aspect of intelligence, although not the only one.

A rational agents has preferences about the state of its environment. What is the rational thing to do depends on the preferences of the agent, what at the agent has perceived so far, what the agent knows about its environment, what its sensors perceive and what the agent deduces from perceptions, what actions the agent is able to take.

In the brokering problem the knowledge is represented by the different proposals that agents perceive, together with the Call for Proposals and the set of constraints and objectives.

3.3. Decision Making. Making a rational agent more intelligent means making it better able to adapt to its environment. It implies more knowledge, better sensors, additional sensors, and enhancing its possible actions, but also better deduction skills.

The Decision Making process, described in detail in the next section, can be seen as an e inference mechanism, that uses perceptions and the knowledge base to deduce which actions to take.

4. The decision process. In our approach, the decision process is modeled as a problem of choosing among many contents or proposals to be retrieved according to the users' need. The process of discovery is performed in different ways depending on the problem and the solution is realized as a multi-agent system.

Multi-agent systems are fundamental enabling technology, especially in situations where mutual interdependencies, dynamic environments, uncertainty, and sophisticated control play a role. They can provide, as stated in [14], robust representational theories and very direct modeling technologies to help us understand large, multi-participant, multi-perspective aggregates.

The decision-making support methodology proposed here is shown in Figure 3.1. First of all it needs to define a knowledge model and its formal representation. It is necessary for the following evaluation and

comparison.

Besides it needs to provide a discovery service to collect the relevant information about the available choices. In particular in the following case studies a semantic discovery has been carried out and a negotiation protocol has been implemented to ask for a proposal to available vendors. Different objectives can be defined and both quantitative and qualitative characterization of available information must be provided.

In fact decision problems are governed by a number of stakeholders with their own objectives and priorities. Finally a strategy to evaluate the optimal decision is needed. The following subsections details each phase of the process shown in Figure 3.1.

4.1. Discovery service. The process of discovery consists of the retrieval of a set of resources, which are are relevant to the user's query, expressed by a set of requirements or a personal profile. The process of discovery is performed in different ways depending on the problem. The aim is the retrieval of all feasible alternatives.

4.2. Evaluation. The process of evaluation consists of assessing an overall evaluation measure to the alternatives in order to provide a ranking of alternative assignments. This overall evaluation is assessed taking into account user's constraints and preferences. The aim is to find the expected utility of each alternative, calculated using an utility function U(x) that depends on the problem.

4.3. Decision Strategy. The process of decision consists of filtering the alternatives with the highest expected utility, that are compliant with user's constraints. The aim is to choose the alternative or the set of alternatives that maximize the utility function.

Results of this phase could still be multiple choices, in fact aim of the decision support is to delegate all the evaluations and decisions, which can be resolved without the user intervention, to agents. Only the remaining uncertainty should be left to the user's attention.

5. The decision problem formulation.

5.1. Agents based modeling. In [33] an agent is defined as a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its objectives. We need to formally define objectives, believes and actions in order to formulate the problem in terms of multi-agent system.

The decision making model can be formally defined by the following items:

- A set of *Believes* $B = \{b_1, \cdots, b_{ne}, \}$, to be used for describing the knowledge in the application context, is described by a domain ontology. The ontology has to be defined for the specific use case.
- The knowledge of agent i at the time t, about the environment and about his own preferences, is described by a set of concepts and individuals belonging to the ontology. It is used to build the query for retrieving the available options to be evaluated in the decision process. For each user i we have a representation of his interest by a subset of preferred believes $P_i \subset B$ and a set of constraints $C(P)$.
- A discovery service for retrieving all available alternatives that are relevant to P_i.
- A set of goals: $G = \{g_1, \cdots, g_{nw}\}$, whose achievement improves the user's utility.
- Let us define $A = \{a_1, \cdots, a_{na}\}$ the set of assets which have been discovered. We need to define a multi-objectives metric for the user utility $U : A \rightarrow u_1(A), \ldots, u_{nw}(A)$, that is the measure of the achievement of each goal. U is a set of functions the defines the evaluation criteria based on which the alternative assets are evaluated;
- A decision maker or group of decision makers will choose the best collection of assets $A' \subset A$ which optimize the user's utility, but it is compliant with user's constraints. The optimal set of assets to be proposed will be $U(G(A')) > U(G(A' \cup a_j))\forall j : a_j \subset A - A'$. It means that any additional asset that is not useful to increase the user's utility must be excluded. Of course both an empty solution, or equivalent ones could be found.

In our multi-agents solution:

- perceptions provide information regarding the context of decision,
- the review process of knowledge updates the user profile and then the requirements for the decision,
- in the discovery phase the goal determines the alternative plans or actions to be evaluated for making the decision.

5.2. Believes and Preferences. In this case study, the set of *Believes* $B = \{b_1, \cdots, b_{ne},\}$ is described using the OCCI taxonomy defined in [17], an interface that offers a uniform access to IaaS resources. Here we define Call For Proposal (CFP) the document to be prepared by the customer to specify his requirements.

The CFP includes the list of resources to be acquired and the rules/policies to be used for defining resource brokering strategies, *i.e.*, prices, availability, etc.

As shown in Fig.5.1, the CFP is composed of two documents. The first one is the *SLA Template* described according to the XML SLA@SOI schema [27]. The *SLA Template* expresses the configuration of resources that are necessary for the user and consists of a set of virtual resources that may include compute, network and storage and can be complemented by the user with other information. In particular, as it is described in [2], the SLA template is composed of *Service Properties*, *Guarantee Terms* and *Terms of Service*.

The second document is the *Broker Policy*, containing a set of rules, to be enforced by the brokering algorithm, in order to choose among the different proposals offered by the Cloud market.

SLA brokering is part of the agent based provisioning service of Cloud Agency. The broker collects a number of proposals described in an vendor agnostic way and chooses the best one(s) according to the brokering rules.

Broker service provided by Cloud Agency is targeted to Cloud developers and deployers, who aim at building their own Cloud Infrastructure.

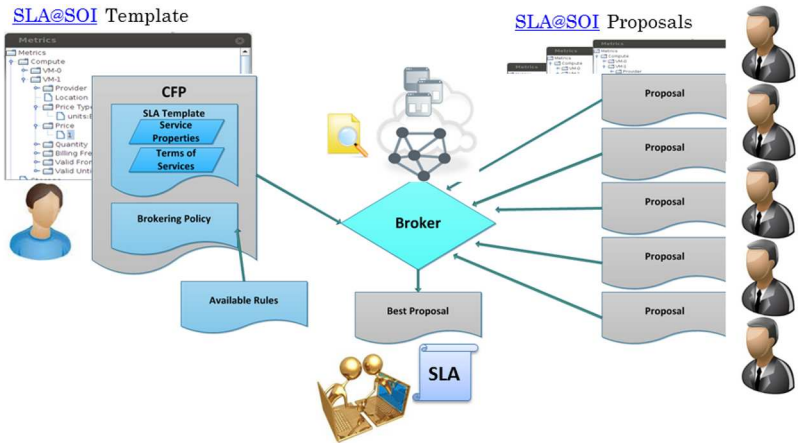

FIG. 5.1. *Broker*

Of course different proposals will come from Cloud Vendors. They will offer different kind of products at different terms. The broker should be able to choose the best one according to the policies specified by the customer such as best price per time unit, maximum amount of memory, service availability and so on.

The set of *Goals* $G = \{g_1, \cdots, g_{nw}\}$ includes the best price, the greatest number of cores, the best accredited provider or the minimum accepted availability. A number of constraints can be also defined to specify the minimal user's requirements.

6. Brokering Model. Our broker finds in a set of SLA proposals $\{P_1, \ldots, P_m\}$ the ones that optimize a multi objectives function and satisfy a number of constraints. Those proposals constitutes the set of *Assets* which have been discovered.

Each proposal is provided by a Cloud vendor complementing the terms of the SLA template $T = \{t_1, \ldots, t_n\}$, received from the customer, with the correspondent offered values $Pj = \{(t_1; v_{j,1}), \ldots, (t_n; v_{j,n})\}$.

In order to evaluate the best proposal the broker uses a set of rules $R = C \cup O$, which can be constraints rules $cr \in C$ or goals rules $or \in O$ rules.

Constraints rules are boolean expressions that represent soft or hard requirements:

$$cr : (t_i, c_i, m_i) \rightarrow [0, 1]$$

t_i is an SLA term, c_i is a boolean expression, and m_i is a float number whose value specifies that the constraints is hard (when 1) or soft (0).

Goals rules assign a score between 0 and 1 to the compliance of the SLA value of the term t_i with the correspondent user's requirements

$$or : (t_i, f_i, o_i) \rightarrow [0, 1]$$

For each goal rule the user has to select a mapping function f_i between the t_i values and the correspondent score, and has to specify if that rule is an explicit ($o_i = true$) or implicit goal ($o_i = false$).

The mapping function change the way used by the user to evaluate that goal. For example logarithmic, linear, and exponential function can be used to define the relevance of that goal according to the value $v_{j,i}$ offered by the provider and the one desired by the customer.

The broker policy will be a subset of rules $R' \subset R$ that is defined for those terms of the SLA template, which are relevant for the user's requirements.

To solve the brokering problem we have to perform the following computation for each received proposal by replacing t_i with the correspondent value $v_{j,i}$:

- $M_j = \prod_{i=1}^{n} \neg(\neg c_i \wedge m_i) \, \forall j = 1, \ldots, m$
  that is used to check if the SLA proposal can be considered as a valid candidate for the SLA , in fact at least a false hard constraint invalidates that offer.
- $Opt_j = \sum_{i=1}^{n} (\neg m_i * cr_i) \, \forall j = 1, \ldots, m$
  evaluates how many soft constraints are met. It can contribute to the evaluation of the proposal.
- $V_j = \sum_{i=1}^{n} (\neg o_i * f_i(v_{j,i})) \, \forall j = 1, \ldots, m$
  represents an overall evaluation for all those terms which have not to be negotiated independently.

All goals rules which have $o_i = true$ will be considered independent goals. In general the best proposals will be the ones which solve the following equations:

$$max_{j=0}^{m}(or_i) : o_i = true \; and \; M_j = 1 \, \forall i = 1, n$$

An additional criteria will be:

$$max_{j=0}^{m}(Opt_j) : M_j = 1$$

All the defined criteria can be grouped if the user set $o_i = false \; \forall i = 1, \ldots, n$, and $m_i = false \, \forall i = 1, \ldots, n$. In this case the result of brokering will be:

$$max_{j=0}^{m}(V_j) : M_j = 1$$

that means the best proposal are the ones with the best overall score.

The policy edited will be translated in a language supported by programs for symbolic computation like Octave, Matlab or Maple and is computed by replacing t_i parameter with actual values of each proposal.

6.1. Goals and Constraints. Goals and constraints are defined by a brokering policy embedded into the CFP. Constraints can be architectural constraints and service level constraints. Hard constraints refer to the fact that the cloud offer must satisfy the required condition, otherwise it is to be excluded. Soft constraints refer to desired requirements that can make a provider preferred with respect to another. The classification of constraints into hard or soft depends on user's need.

Explicit objectives can be the maximization or minimization of some parameters (e.g.: memory and price). Implicit objectives are automatically evaluated by the broker. An example is the amount of soft constraints that have been satisfied.

The rules can be defined selecting the SLA parameters and setting the required options using a friendly graphic interface.

Examples of constraints rules, shown in Table 6.1 are:

- exact matches,
- value in a set,
- greater/less then,
- value in a range.

TABLE 6.1
Rules Types

Rule's Name	**Value Type**	**Boolean Expression**
Exact Match	Numerical & Non Numerical	$t_i = s$
Value in a Set	Numerical & Non Numerical	$t_i \in S$
Greater then	Numerical	$t_i > s$
Less then	Numerical	$t_i < s$
Value in a Range	Numerical	$t_i \in R$

Of course not every constraint can be applied to any SLA parameters.

Some among the same parameters can be considered as input of objectives functions to be optimized [18]. There may be no one single goal for the optimization, and no single optimal solution. For example, given a set of constraints, a goal could be the minimization of the cost, maximizing the memory.

In this case, a multi-objective approach should be adopted and one of the solutions on the Pareto front (that is a set of all those solutions that are considered to be optimal in multi-criteria optimization) should be chosen. To choose one of the solutions on the Pareto front, a posteriori approaches is used that deliver to the user the set of Pareto-optimal solutions among which the user will choose the preferred one.

In order to simplify the usage of the brokering service we allow for grouping multiple objectives according to the kind of SLA parameter: *Service Properties*, *Terms of Services* or *Service Levels*. We also define the *Provider Reputation* as an additional brokering parameter, that is out of the SLA Template, but it is known to the broker.

To compute the overall score we map the domain of each SLA parameter to $[0, 1] \subset \mathcal{R}$ and we allow to assign a percentage relevance to each category.

7. Brokering As A Services.

7.1. Cloud Agency Brokering AAS requirements. Cloud Agency, presented in details in [30], is a Multi-Agent System that complements the common management functionalities which are currently provided by Private and Public Infrastructure as a Service (IAAS) with new advanced services, by implementing a Vendor Agnostic layer.

The Provisioning service of Cloud Agency implements the FIPA Contract-Net protocol described in [26]. It is a minor modification of the original contract net IP pattern in that it adds rejection and confirmation communicative acts.

In the contract net IP, one agent (the Initiator) takes the role of manager, which wishes to have some tasks performed by one or more other agents (the Participants) and further wishes to optimize a function that characterizes the task. This characteristic is commonly expressed as the price per time unit, in some domain specific way, but could also be soonest time to completion, fair distribution of tasks, etc.

For a given task, any number of Participants may respond with a proposal; the rest must refuse. Negotiations then continue with the Participants that returned valid proposals. For each received CFP Cloud Agency creates a broker that searches for vendors that can offer resources with the required QoS (Quality of services).

Cloud Agency implements a single-user service at platform level that can be used to provide autonomic capability to Cloud applications over IaaS infrastructures.

At Application as a Service level we have to overcome the performance limitations of the mOSAIC Cloud Agency solution as it has been developed [31], starting from the identification of new requirements.

First of all it needs to identify the issues introduced by the new scenario, but also to take in consideration the available technologies for designing and implementing a new engineered solution. About the issues we need to take into account the number of service users who can access contemporary the service by multiple requests.

We have to consider that in a service oriented context, not only human users, but also applications and robots will be able to invoke the service producing different kind of workloads. The new workload can vary in

dimension, but also it can dynamically change during the day, with regular or unforeseeable bursts on special periods. For this reason we have to share the workload over a distributed computing infrastructure, and we need to grant that both the infrastructure and the application will scale dynamically.

7.2. Architecture design and implementations. The Cloud technology is a promising solution to build a scalable computing infrastructure, but it needs to re-design the agents in order to let them exploit such an elastic computing model.

In fact while the Cloud allows to scale the computing resources according to the measured workload improving also the utilization, the application must be able to reconfigure itself autonomically to keep the QoS level above the desired threshold. Hence we are going to build a distributed Cloud broker over a Cloud infrastructure.

Not only performance issues should be addressed, but also reliability and availability when the service execute over such kind of distributed platform, due to lack of control both of the network and of the compute utility.

Hence, the main assumption we will take as strict requirement here will be the design of stateless and asynchronous scheduling of agents that will be able to run on every available computing resources.

This solution delegates the evaluation of a single proposal to any idle broker within a pool of agents, who are waiting for the evaluation of a proposal and update an eventual optimal solution. This design choice allows for the easy distribution of the workload using a task parallel programming model. On the other hand the front-end will be implemented by a service interface that accepts synchronous requests. It stores the new pending tasks to be handled by pools of asynchronous working agents, and returns the current status of service elaboration.

The service architecture is shown in Figure 7.1. In Figure 7.1, in the upper left corner, we can see many service instances that receive requests from end users and access in-memory shared information. The in-memory session manager will allow for the exploitation of elastic capability of the Cloud infrastructure. The warm copy of the session manager allows for improving reliability. As it is shown in Figure 7.1 agents will implement the back-end of the new agency, that is completely relieved of handling interactions with clients. Cloud storages (database ad queues) keep persistent information of the distributed applications and implement communication channel between synchronous front-end handlers and back-end workers. This design choice allows for the easy distribution of the workload using a task parallel programming model.

Stateless agents take new problems by a common bag of task, execute wherever there are available computing resources, and update the computing results if they complete successfully. Reliability is addressed by re-scheduling of unsolved problems which remain in the bag because of any failures or delay.

The vendors agent sign up to the CFP_QUEUE to receive the cfps received from users. Each of them submits its proposal to the PROPOSALS_QUEUE. Idle brokers are waiting for proposals. A single proposal is dispatched to one broker that executes its matching with the correspondent CFP for evaluation purpose. The matching results is updated into the SLA_QUEUE if it belongs to the Pareto front of optimal solutions. Brokering results are stored also into an in-memory session, together with information of each related user's request.

Users are provided with the web interface shown in Figure 7.2. It allows for composing the CFP and listing, for each session, the best SLAs according to different brokering objectives. Users log into a web page, the web page makes a request to a REST service and submits the call for proposals.

The Call for Proposal are included, along with information of the session and the user, in CFP queue. The characteristic of the CFP queue is that all consumers that join the queue, receive all the submitted CFPs.

Apache Tomcat has been used as web and application server to run the web service at front-end. It has been specifically configured for working with the Terracotta Framework for the transparent distribution and sharing of web sessions.

Jersey API have been used to implement the RESTFull service that provides methods for authentication, CFP submission and SLA retrieval. Meanwhile their requests are pending, users can wait for the result of brokering or may poll periodically to get the status of their request. When one of the brokers has found a feasible proposal for that request the current results are updated. The completion of brokering is notified when there are no more proposal candidate to optimize the user's query to be evaluated.

The RDBMS Mysql databases is used as persistent storage of CFPs and to SLAs. ActiveMQ has been used as queue services for communication and synchronization.

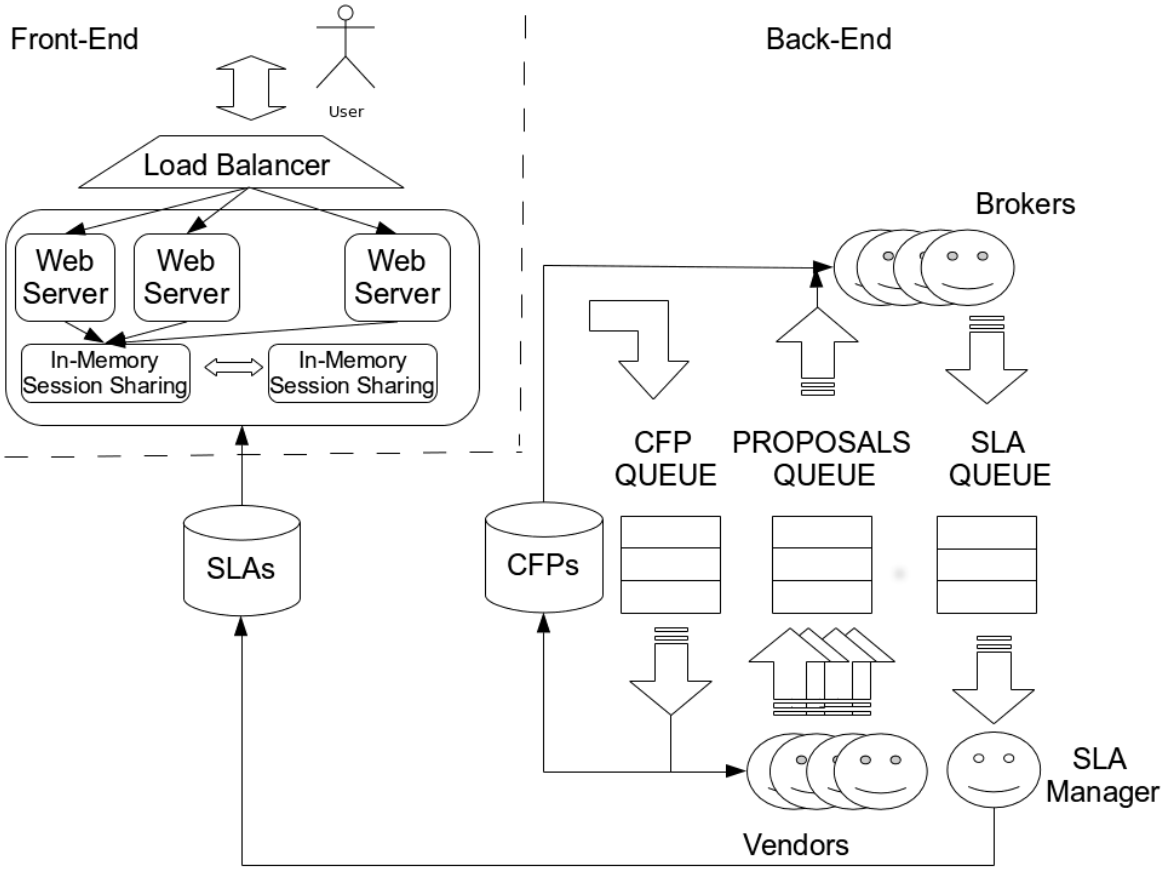

FIG. 7.1. *Distributed architecture*

In this solution, on the arrival of any Call for Proposal worker agents will cooperate to compare all the proposal candidate to satisfy the brokering policy. Each agent continuously will keep from a queues the next proposal to be evaluated, till when there are not more waiting requests. On their own users will be able to query the system about the status of their requests and the intermediate results.

7.3. Experimental results and discussion. In order to evaluate performances of the proposed approach we set up the following testbed. A Linux physical machine hosts the ActiveMQ 5.6 service, with a Topic, named CFP_QUEUE, that receives CFPs from concurrent clients, which run on a different physical machine in the same 1GB Ethernet local network. The server (H1) is 64bit Intel Core T M 2 Quad Processor Q9300 (6M Cache, 2.50 GHz, 1333 MHz FSB) with 4GB RAM. Oracle Java7 is the runtime environment.

Concurrent clients send 10 CFPs, each one, according to a Poisson process with different mean time of arrivals. Vendors run on the server and wait for incoming CFPs. All vendors get the same CFP and generate their proposal, which is sent to the $PROPOSALS\_QUEUE$.

In a first scenario all brokers run on the server itself. They receive a different proposal from the QUEUE and evaluate the compliance with the correspondent CFP. The result is sent to an SLA_QUEUE from which only the best ones are notified to the clients.

In different scenario, in order to improve this parameter we investigated the possibility to offload part of the workload by a Cloud Infrastructure.

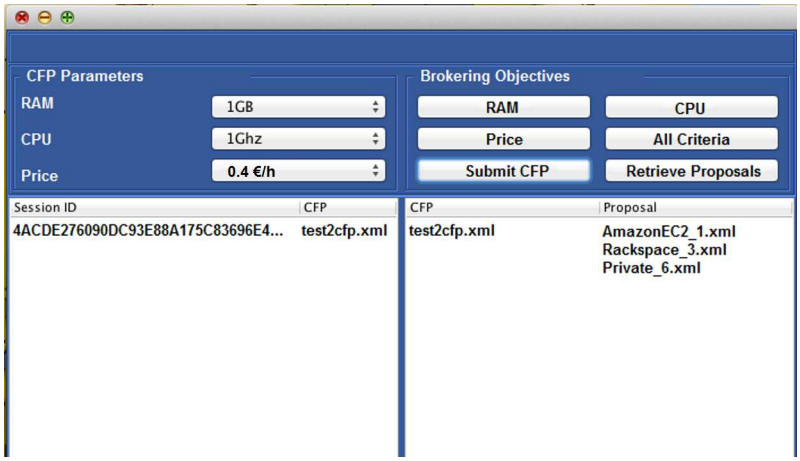

FIG. 7.2. *Web GUI*

We used an OpenStack installation in the same local network. This private cloud provided a Linux virtual machines (H2). The Linux OS sees just one processor with a 64bit virtual Intel Core 2 Duo P9xxx (Penryn Class Core 2) 2.5GHz with 2K L1 cache and 2GB RAM.

We evaluated the performance of such configurations changing the number of clients, the number of vendors and the number of brokers.

We observed a increasing throughput in terms of number proposals per second evaluated by brokers when the mean time of arrivals decreases and the number of brokers increases.

A drop of the throughput has been observed when the number of brokers running on a machine is greater than the number of cores, both on the server and on the working machines, and when the system is overloaded by a huge number of proposals to be evaluated, which depends wither on the number of vendors and on the mean time of CFPs arrivals.

In the first case the time of proposals evaluation increases due to the scheduling overhead. In the second case a noticeable effect is above all the average waiting time of a proposal into the queue, that affects directly the response time of the system and service level perceived by the client.

In Table 7.1 we show the mean enqueued time of a proposal in the case of 8 clients sending 10 CFPs with a mean time of arrivals of 500 ms. In all the case we have 8 vendors, that means 640 proposals to be evaluated.

Due to the unbalance between the computational power of H1 and H2 the queue service is able to dispatch a limited number of proposals to the working machine spawned in the Cloud and we have not a relevant benefit. We expect the with more homogeneous machines and with an greater number of working nodes we will get better improvement.

8. Conclusion. In this paper we presented a methodology that uses Multi Criteria Decision-Aid for designing and developing multi-agents solution of problems, whose complexity is characterized by different criteria to be evaluated and compared independently and according to the user's/agent's preferences.

We applied the proposed methodology to solve brokering problem of Cloud service with multiple objective and constraints set by the users' requirements. The proposed Brokering As A Service application has been built and executed on a distributed computing infrastructure that uses Cloud resources as working machines and asynchronous brokers for evaluating independent proposals from different vendors.

To perform an exhaustive search of the optimal solutions we assumed that the resulting SLA cannot be composed with offers from different vendors to dominate the complexity. We evaluated the performance changing the mean time of arrivals of CFPs from multiple clients and the number of vendors.

Future works will take in consideration larger problem by service compositions, using bigger computing infrastructure and heuristics for the computation of sub-optimal solution within the required time limits.

Furthermore we aim at advancing the current results by extending the proposed methodology to support multiple conflicting decisions in many-to-many negotiation protocols and by Game Theory techniques.

TABLE 7.1
Proposals mean enqueued time

Client	H1	H2	Mean Enqueued Time
8c	8v+1b		428
8c	8v+2b		180
8c	8v+4b		170
8c	8v+8b		171
8c	8v	1b	1430
8c	8v	2b	920
8c	8v	4b	11296
8c	8v+4b	1b	165.578
8c	8v+4b	2b	160
8c	8v+4b	4b	167

REFERENCES

[1] A. AMATO, G. CRETELLA, B. D. MARTINO, AND S. VENTICINQUE, *Semantic and agent technologies for cloud vendor agnostic resource brokering*, in AINA Workshops, 2013, pp. 1253–1258.

[2] A. AMATO, L. LICCARDO, M. RAK, AND S. VENTICINQUE, *Sla negotiation and brokering for sky computing*, in CLOSER, 2012, pp. 611–620.

[3] A. AMATO, B. D. MARTINO, AND V. SALVATORE, *Cloud brokering as a service*, in The 8th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, USA, December 28-30 2013, IEEE Computer Society, pp. 9–16.

[4] A. AMATO, L. TASQUIER, AND A. COPIE, *Vendor agents for iaas cloud interoperability*, in Intelligent Distributed Computing VI, G. Fortino, C. Badica, M. Malgeri, and R. Unland, eds., vol. 446 of Studies in Computational Intelligence, Springer Berlin Heidelberg, 2013, pp. 271–280.

[5] A. AMATO AND S. VENTICINQUE, *Multi-objective decision support for brokering of cloud sla*, in AINA Workshops, 2013, pp. 1241–1246.

[6] F. ANDRÈ, M. CARDENETE, AND C. ROMERO, *Basic aspects of the multiple criteria decision making paradigm*, in Designing Public Policies, vol. 642 of Lecture Notes in Economics and Mathematical Systems, Springer Berlin Heidelberg, 2010, pp. 33–53.

[7] H. ARMESH, *Decision making*, in Proceedings of the 12th International Business Research Conference, Melbourne, Australia, 2010, World Business Institute, pp. 11–21.

[8] R. BUYYA, R. RANJAN, AND R. N. CALHEIROS, *Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services*, in ICA3PP (1), 2010, pp. 13–31.

[9] S. J. CHEN, C. L. HWANG, AND F. P. HWANG, *Fuzzy multiple attribute decision making: Methods and applications in collaboration with dr. frank p. hwang*, (1992).

[10] P. DELIAS AND N. F. MATSATSINIS, *The multiple criteria paradigm as a background for agent methodologies*, in 8th Annual International Workshop 'Engineering Societies in the Agents World', 2007, pp. 227–237.

[11] B. DI MARTINO, D. PETCU, R. COSSU, P. GONCALVES, T. MÁHR, AND M. LOICHATE, *Building a mosaic of clouds*, in Proceedings of the 2010 conference on Parallel processing, Euro-Par 2010, Berlin, Heidelberg, 2011, Springer-Verlag, pp. 571–578.

[12] F. DIGNUM, D. MORLEY, E. SONENBERG, AND L. CAVEDON, *Towards socially sophisticated bdi agents*, in MultiAgent Systems, 2000. Proceedings. Fourth International Conference on, 2000, pp. 111–118.

[13] M. DOUMPOS AND E. GRIGOROUDIS, *Multicriteria Decision Aid and Artificial Intelligence: Links, Theory and Applications*, Wiley, 2013.

[14] L. GASSER, *Mas infrastructure: Definitions, needs and prospects*, in Revised Papers from the International Workshop on Infrastructure for Multi-Agent Systems: Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems, London, UK, UK, 2001, Springer-Verlag, pp. 1–11.

[15] M. P. GEORGEFF, B. PELL, M. E. POLLACK, M. TAMBE, AND M. WOOLDRIDGE, *The belief-desire-intention model of agency*, in Proceedings of the 5th International Workshop on Intelligent Agents V, Agent Theories, Architectures, and Languages, ATAL '98, London, UK, UK, 1999, Springer-Verlag, pp. 1–10.

[16] N. R. JENNINGS, K. SYCARA, AND M. WOOLDRIDGE, *A roadmap of agent research and development*, Autonomous Agents and Multi-Agent Systems, 1 (1998), pp. 7–38.

[17] T. METSCH, A. EDMONDS, AND R. NYREN, *Open cloud computing interface – core and models*, in Standards Track, no. GFD-R in The Open Grid Forum Document Series, Muncie (IN), 2011, The Open Grid Forum Document Series.

[18] F. MOSCATO, R. AVERSA, AND A. AMATO, *Describing cloud use case in metamorp(h)osy*, in CISIS, 2012, pp. 793–798.

[19] S. K. NAIR, S. PORWAL, T. DIMITRAKOS, A. J. FERRER, J. TORDSSON, T. SHARIF, C. SHERIDAN, M. RAJARAJAN, AND A. U.

Khan, *Towards secure cloud bursting, brokerage and aggregation*, in Proceedings of the 2010 Eighth IEEE European Conference on Web Services, ECOWS '10, Washington, DC, USA, 2010, IEEE Computer Society, pp. 189–196.
[20] NIST, *Nist cloud computing reference architecture - special publication 500-292*. Available at `http://www.nist.gov/`.
[21] D. Petcu, C. Craciun, M. Neagul, S. Panica, B. D. Martino, S. Venticinque, M. Rak, and R. Aversa, *Architecturing a sky computing platform*, in ServiceWave Workshops, 2010, pp. 1–13.
[22] B. Roy, *Multicriteria Methodology for Decision Aiding*, vol. 12 of Nonconvex Optimization and Its Applications, Springer, Formerly Kluwer Academic Publishers, Dordrecht, Boston, London, 1996. Translator: Mark R. McCord.
[23] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.
[24] K. M. Sim, *Towards complex negotiation for cloud economy*, in 5th International Conference on Advances in Grid and Pervasive Computing (GPC 2010), 2010, pp. 395–406.
[25] R. Slotboom, *Mobile agents as a distributed application architecture*, 2000.
[26] F. I. P. A. Technical report, *Fipa contract net interaction protocol*, 2002. Available at `http://www.fipa.org`.
[27] W. Theilmann, *Slasoi*, 2011. Available at `http://sla-at-soi.eu/`.
[28] E. Triantaphyllou, B. Shu, S. Nieto Sanchez, and T. Ray, *Multi-criteria decision making: An operations research approach*, In: Encyclopedia of Electrical and Electronics Engineering, vol. 15 (1999), pp. 175–186.
[29] P. Tučník, J. Kožaný, and V. Srovnal, *Multicriterial decision-making in multiagent systems*, in Computational Science – ICCS 2006, V. Alexandrov, G. Albada, P. Sloot, and J. Dongarra, eds., vol. 3993 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2006, pp. 711–718.
[30] S. Venticinque, *User-centric infrastructure as a service by cloud agency*, MULTIAGENT AND GRID SYSTEMS, 9 (2013), pp. 157–159.
[31] S. Venticinque, R. Aversa, B. Di Martino, M. Rak, and D. Petcu, *A cloud agency for SLA negotiation and management*, vol. 6586 LNCS of Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Springer, 2011, pp. 587–594.
[32] M. Wooldridge, *Intelligent agents: The key concepts*, in Proceedings of the 9th ECCAI-ACAI/EASSS 2001, AEMAS 2001, HoloMAS 2001 on Multi-Agent-Systems and Applications II-Selected Revised Papers, London, UK, UK, 2002, Springer-Verlag, pp. 3–43.
[33] M. Wooldridge, *An Introduction to MultiAgent Systems*, Wiley Publishing, USA, 2nd ed., 2009.
[34] H.-J. Zimmermann, *Fuzzy set theory and its applications, second edition*, (1991).
[35] R. Šperka and K. Slaninova, *The usability of agent-based simulation in decision support system of e-commerce architecture*, I.J. Information Engineering and Electronic Business, 4 (2012), pp. 10–17.

Edited by: Teodor Florin Fortiş
Received: Mar 3, 2014
Accepted: Apr 16, 2014