Expert Systems with Applications xxx (2012) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Model for assigning roles automatically in egovernment virtual organizations

Carolina Zato, Juan F. De Paz *, Ana de Luis, Javier Bajo, Juan M. Corchado

Departamento Informática y Automática, University of Salamanca, Plaza de la Merced s/n, 37008 Salamanca, Spain

ARTICLE INFO

Keywords:
Multi-agent systems
Virtual organizations
Queuing theory
Task planning
E-goverment

ABSTRACT

When working in conjunction with multiagent systems, virtual organizations attempt to simulate the functions and interactions of entities in different environments. Recent studies have addressed the problem of assigning roles to agents that form part of the organization, and incorporating new agents to carry out certain tasks. However, these studies are limited to defining the norms and rules that determine the behavior of the organization. The present study proposes a virtual organization model for egovernment environments to assign resources and minimize the required personnel by forecasting workloads. To this end, a neural network, queuing theory, and CBR are used to obtain an efficient distribution. Queuing theory can establish the number of agents with a specific role that are necessary to maximize profits, while the network distributes roles among agents according to their respective efficiency. The final part of the paper is focused on validating the plan developed inside a case study centered on e-government in order to obtain empirical results.

1. Introduction

Various studies have explored the concept of planning in virtual enterprises or organizations (Camarinha-Matos & Afsarmanesh, 2007; Fenga & Yamashiro, 2006; Verter & Dincer, 1992; Wu, Fuh, & Nee, 2002). Nevertheless, these studies have focused on defining the objective parameters and functions of problems, which inhibits their broader application. Furthermore, these systems neglect to address potential demands and thus provide plans based on the current workload at a given time. Other studies seek to optimize the allocation of resources to maximize efficiency and minimize costs by means of different heuristic or precise optimization techniques (Fenga & Yamashiro, 2006; Verter & Dincer, 1992; Wu et al., 2002). The main problem in these cases stems from the dynamic aspect of work scenarios and the difficulty in finding a balance between time spent on planning and time spent on implementing plans, which is the key to adapting to the needs of organizations such as these that require frequent replanning. We therefore propose the need to create a system that is capable of carrying out an efficient planning system not only for current and existing work, but for the prediction of future work as well.

Planning strategies are commonly based on minimizing objective and multiobjective functions, and matching offers and demands (Camarinha-Matos & Afsarmanesh, 2007). The problems that arise from objective functions are resolved by exact or heuristic means. The exact methods, such as linear programming (Kabadi & Punnen, 2008), nonlinear programming (Shi, Lu, & Zhang, 2005; Wan, Yang, & Izquierdo, 2009) and graph theory, ensure that an optimal solution is provided in execution time according to the number of existing variables. Early research in virtual organizations was based on optimization through whole linear programming (Fenga & Yamashiro, 2006; Verter & Dincer, 1992). However, techniques such as linear or nonlinear programming are not applicable to NP-Hard problems. Moreover, it is inherently difficult for these techniques to define constraints and objective functions because the objective function must present all of the existing combinations in order to define the variables, and this number of combinations can be quite high. As a result, it is necessary to either resort to heuristics to solve optimization problems (Wu et al., 2002) in which the parameters of the objective function are calculated, or to combine heuristic techniques with linear programming to solve these types of problems (Verter & Dincer, 1992). It is advisable to use metaheuristic techniques (Kim & Park, 2004; Kolonko, 2009; Porto, Kitajima, & Ribeiro, 1996; Taillard, 1994) to solve these types of problems, as they can obtain efficient solutions in reasonable execution times.

This study proposes a multiagent based Virtual Organization (VO) model with planning and resource distribution capabilities, as well as the ability to estimate work demands in order to both determine the number of resources needed and, according to the number of demands, to distribute work resources in such a way that maximizes profits and minimizes delays. The planning system will be developed according to a case-based reasoning (CBR)

* Corresponding author. Tel.: +34 923294400x1926.
E-mail addresses: carol_zato@usal.es (C. Zato), fcofds@usal.es (J.F. De Paz), adeluis@usal.es (A. de Luis), jbajope@usal.es (J. Bajo), corchado@usal.es (J.M. Corchado).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2012.01.185

system (Kolodner, 1993) which can be integrated into the various stages of reasoning techniques in order to estimate resources. This estimation is based on queuing theory and planning, and uses a variant of a multilayer perceptron that incorporates unsupervised learning. The planning mechanism is applied to virtual organizations (Boella, Hulstijn, & van der Torre, 2005; Esteva, Rodríguez, Sierra, Garcia, & Arcos, 2001; Markus, Ingo, Josef, & Helmut, 2010; Rocha & Oliveira, 1999) of agents (Durfee, Lesser, & Corkill, 1989) to simulate the behaviour of the organization in its planning and allocation of work for the given e-government case study. Due to the high number of procedures and users involved in this type of entity, it is essential to have a good planning system that can ensure the work is carried out successfully and in a timely manner, and that the staff allocated to the project is the minimal number required to successfully meet future challenges.

Section 2 of this paper presents the state of the art for the different concepts used: multiagent systems, virtual organizations, and task planning methods. Section 3 presents the proposed model and includes solutions for a dynamic distribution of roles and subsequent task assignments. The solution is explained in detail in Section 4, which also includes the case study and corresponding results. Finally, Section 5 presents our conclusions and future work.

2. Virtual organizations and multi-agent architectures

The technology of dynamic agent organizations that auto-adjust to obtain advantages from their environment is more than suitable for coping with the development of this type of system. These organizations could appear in emergent or dynamic agent societies, such as grid domains, peer-to-peer networks or other contexts where agents dynamically group together to offer compound services. However, although technology, on a theoretical level, seems to be able to allow the development of this new kind of system, it is still necessary to investigate theories, models, mechanisms, methods and tools in order to develop systems with reorganization capabilities that provide them with the ability to adapt to environmental changes.

Multi-agent systems are characterized by being autonomous, reactive, proactive, socially skilled, etc. (Wooldridge & Jennings, 1995) and therefore are perfectly suited for creating organizational models in which each agent can send and receive messages with an individual, organization or actor in the real society that is being modelled (David, Marietto, Sichman, & Coelho, 2004). Additionally, the interaction between agents can correspond to interactions existing in the real world (David, Marietto, Sichman, & Coelho, 2004). There are many agent-based social simulation models that try to analyze different social phenomena (David, Marietto, Sichman, & Coelho, 2004; Epstein & Axtell, 1996). Schelling (1978) carried out the first agent-based social simulation in which each person was represented by an agent and the interaction between these agents represented relevant social processes. Nevertheless, there is still much work to be done, especially in the field of automated re-organization of virtual organizations (David, Marietto, Sichman, & Coelho, 2004).

The open MAS (Bajo, Corchado, Botti, & Ossowski, 2009; Zambonelli, Jennings, & Wooldridge, 2003) should allow the participation of heterogeneous agents, which change over time, with architectures and even with different languages. For this reason, we cannot rely on agent behavior when it is necessary to establish controls on the basis of norms or social rules that can affect the organizational architecture and the agents residing within it. This is one of the reasons that encourage the use of virtual organizations (VO). A VO (Esteva, Rodríguez, Sierra, Garcia, & Arcos, 2001) is an open system designed for grouping, for the collaboration of heterogeneous entities, and for when there is a separation between form and function that defines their behavior. Early works, such as that of Rocha and Oliveira (1999) propose initial versions of open MAS based on the definition of the Market agent, which assumes the role of coordinator and selects the agents that belong to the organization according to a set of restrictions. Other more recent works (Markus, Ingo, Josef, & Helmut, 2010) provide a 3D simulation of the behavior of Open MAS, although its operation continues to be based on the definition of restrictions that are validated by the SMA Ameli prior to executing a transition. Other trends such as (Boella, Hulstijn, & van der Torre, 2005) define standards of cooperation between agents according to their interaction, and use BDI models to establish behavior. Camarinha-Matos and Afsarmanesh (2007) raised the issue of coordination within virtual organizations and, since it is no longer a question of problems that are limited to a simple matching between needs and resources but one that involves multiple variables, suggested that these systems be used to assist in the decision-making process.

In recent years, coordination mechanisms with agents have been proposed in different ways (Jennings & Wooldridge, 1998), the most common of which are the mediated methods. The intermediary methods play an important role in artificial societies in a way similar to what occurs in human societies.

There are many coordinator models, of which the two primary and opposing methods are global and individual coordination. In global coordination, the MAS determines and plans the actions of the agents, while in the individual coordination, the MAS completes the autonomy of the agents.

The present study proposes a global planning model developed by THOMAS that can carry out a global coordination in an organization. THOMAS (Carrascosa et al., 2009) arose from the need to support open multi-agent systems in virtual organizations. The architecture presents the infrastructure needed to develop the concepts of agents, and applies splitting, abstraction and organization techniques while taking all requirements into account.

The use of THOMAS implies that the organizations are composed of Hierarchical Organizational Units in which a supervisor agent has control over all other members, coordinates the tasks, and centralizes the planning and decision process. The THOMAS (Carrascosa et al., 2009) platform additionally provides new adaptation methods that allow the structure to be changed so that it may adapt to the external changes.

Approaches towards the integration of multi-agent systems with SOA and Web Service have recently been explored (Ardissono, Petrone, & Segnan, 2004). 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. Oliva, Natali, Ricci, and Viroli (2008) 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). Shafiq, Ding, and Fensel (2006) propose a gateway that allows interoperability between Web Services and multi-agent systems. Liu (2007) propose a multi-agent architecture to develop inter-enterprise cooperation systems using SOA and Web Service components and communication protocols. Walton (2006) present a technique to build multi-agent systems using Web Services, defining a language to represent the dialogs among agents. There are also frameworks, such as Sun's Jini and IBM's WebSphere, which provide several tools to develop SOA-based systems. Jini uses Java technology to develop distributed and adaptive systems over dynamic environments. Rigole, Holvoet, and Berbers (2002) have used Jini to create agents on demand in a home automation system, where each agent is defined as a service in the network. WebSphere provides tools for several operating systems and programming languages. However, the systems developed using these

Please cite this article in press as: Zato, C., et al. Model for assigning roles automatically in egovernment virtual organizations. *Expert Systems with Applications* (2012), doi:10.1016/j.eswa.2012.01.185

frameworks are not open at all because the framework is closed and services and applications must be programmed using a specific programming language that supports their respective proprietary APIs (Application Programming Interface).

3. Planification models

In its broadest sense, the problem of planning consists of establishing an appropriate order of execution to a set of activities according to specific critiera such as efficiency, quality, time, cost, etc. Recent interest in the field of business and economics, and the corresponding need to provide answers to the problems of planning, has led many researchers to propose new planning mechanisms using Artificial Intelligence (Tupia, 2004). One such example is the task scheduler, which can be defined as a set of tasks that must be executed by a group of heterogeneous entities, and whose purpose it is to find an order and distribution that minimizes the number of available resources and can meet the restrictions of each individual task.

When looking to solve planning problems, there are generally two available options to choose from: exact methods and approximate methods. The first is an optimal method that consists of calculating the exact solution; this may involve lengthy computations, which make this approach impractical. The second alternative involves the use of heuristic methods to calculate a near-optimal solution; this option is less time-consuming, which justifies its use.

The nature of the problems is combinatorial and therefore the time required to find the optimal solution grows exponentially with the number of tasks considered. These problems fall within the scope of combinatorial optimization and can be considered, at best, NP (non-deterministic polynomial) problems. These problems do not have a polynomial algorithm that resolves them, but they do have an algorithm that provides a non-optimal solution, although it is very time-consuming for the few instances it provides. However, the majority of the combinatorial optimization problems belong to NP-hard problems (a subset of NP problems). This situation justifies the application of heuristic algorithms for the search of solution in reasonable time.

Heuristics is one of several procedures of approximation. Although the solutions provided by a heuristic approach are generally quite good, heuristics cannot guarantee that the optimal solution for a problem will actually be found. Furthermore, in most cases it is difficult to adapt these algorithms to a problem that is only slightly different from those for which the algorithm was created in the first place. One example of a heuristic algorithm is the voracious algorithm (AVM), which selects the optimal option during each step of a problem solving process with the goal of finding an optimal general solution. This algorithm is also referred to as greedy myopic: greedy because it always selects the best candidate to form part of the solution; and myopic because the selection is unique and unmodifiable, as it does not analyze beyond the effects of having selected an element as part of the solution. Some examples of these applications can be found in these studies (Tupia & Mauricio, 2004).

The heuristic algorithms that are easily adapted to other problems are called meta-heuristics. Meta-heuristics are intelligent strategies for designing, improving and optimizing general heuristic procedures and providing them with a higher performance compared to traditional heuristics. The most common meta-heuristics algorithms are commented below.

3.1. GRASP algorithm

The term GRASP Algorithm is an acronym for Greedy, because of the criteria used by the algorithm in choosing the optimal value after each step of selecting a candidate; Randomized, because the algorithm randomly chooses a candidate from the list it has obtained; Adaptive, because it is capable of adapting to different application contexts or relevant modifications of the model; and Search Procedure, because it searches within a space or neighborhood, randomly evaluating a set of possible solutions.

While the AVM algorithm focuses on selecting the candidate with the best value at a given time, the GRASP algorithm broadens the restriction to select not necessarily the best solution, but one within a given set of values. Each possible solution is evaluated by an objective function, thus ensuring the selection of a better solution than that provided by the voracious algorithm. These values are not necessarily local optimal values, but they are the primary means of finding one by using a local search. Examples of the application of this algorithm in a planning process can be found in these studies (Andres, Miralles, & Pastor, 2008; Binato, Hery, Loewenstern, & Resende, 2000; Kim & Park, 2004).

3.2. Tabu search

This algorithm performs a local search based on the notion that a specific solution can be improved by applying minor changes. For each solution a set of neighbor solutions are calculated; the algorithm then selects the best solution to continue with the next step. The algorithm stores a search history to prevent the iteration from entering into a loop and becoming confined to local values. There are usually two types of memory: the short term memory that stores the most recent searches; and the long term memory that contains older searches. The short term memory consists of a tabu list that stores information regarding the most recently accessed solutions, so that, any solutions matching a case in the tabu list are summarily rejected. The following publications reviewed some task planning cases in which the application of these algorithms had an effective result (Chambers & Barnes, 1996; Porto, Kitajima, & Ribeiro, 1996; Taillard, 1994).

3.3. Simulated annealing

Simulated Annealing is based on a close analogy between the process of thermodynamic physics and that of a combinatorial optimization problem. The traditional local search algorithms start with an initial solution that is gradually transformed into other solutions that are in turn improved by the addition of minor changes or mutations. If the new solution is better than the previous result, the solution is updated and the system repeats the process until it is no longer possible to obtain a better solution. In this way, the search ends with a local optimum that is not necessarily a global optimum. One way of avoiding this type of problem is to direct the result to a worse solution. However, if the search is in fact leading to a good solution, these kinds of escape movements must be very carefully applied because the system could be leading to a global minimum. Simulated Annealing must be carried out with a probability function that reduces the probability of the escape movements leading to worse solutions during the search, which would in turn assume that the solution is closer to a global optimum.. Some studies related to the use of this method for task planning can be found in Kolonko (2009), Steinhofel, Albrecht, and Wong (1999) and van Laarhoven, Aarts, and Lenstra (2002).

3.4. Genetic algorithms

The general purpose of evolutionary algorithms is to guide the stochastical search in producing a set of structures from which the most appropriate solutions are selected iteratively and a quasi-optimal solution is ultimately found. Unlike other methods, there is not just one solution, but an entire set of solutions for each

Please cite this article in press as: Zato, C., et al. Model for assigning roles automatically in egovernment virtual organizations. *Expert Systems with Applications* (2012), doi:10.1016/j.eswa.2012.01.185

iteration of the algorithm. These methods are based on generating, selecting, combining, and replacing solutions. Because they manage several solutions at a time during the search process, the execution time tends to be greater than other meta-heuristic algorithms. In general terms, a genetic algorithm consists of a population of solutions coded in the same way as chromosomes. The chromosomes have a fitness value that quantifies the 'goodness' of a solution to the problem. The number of opportunities for reproduction will be determined according to this value. There is, furthermore, some probability that these chromosomes will mutate. The main advantages of genetic algorithms include their ease in adapting to both general and specific problems, their broad theoretical base, which can easily hybrid with other paradigms, their easy implementation, and the number of empirical tests that provide specific operators. These algorithms have been used in several planning problems, such as those found in Lee, Wang, and Miao (2008), Shi, Lu, and Zhang (2005) and Yu and Buyya (2006).

In recent years, many task planning theories have been proposed with varying rates of success. In Seda (2007) the authors present an interesting mathematical model applied to the problem of task planning. A review of different comparative studies (Pacheco & Casado, 2003; Rodríguez, 2003) on the efficiency of various meta-heuristics indicates that no single technique stands out clearly above the rest. In the present study, we have opted to apply a neural network in an attempt to reduce the execution time below that of the previously studied hetaheuristic models.

4. Proposed architecture

The proposed architecture is an organization model for egovernment environments that incorporates agents with capabilities for automatically generating work plans and distributing tasks. The core of the system is a novel plan-based planning mechanism that interacts with a BDI model through the implementation of web services, which allows for self-adaptive capabilities in different environments. The system additionally provides communication mechanisms to facilitate integration with the SOA architecture.

The proposed model was designed to develop a planning mechanism to coordinate the agents located inside the virtual organizations. To this end, a set of roles and activities are defined according to these needs. Agents with certain roles act as service coordinators and controlers; the services are responsible for implementing the behaviour of the agents and processing the information, which facilitates tasks involving replication and modulation. In the proposed system, the roles in the platform are hierarchical: the higher organizational layer contains roles from the platforms responsible for the task distribution functionalities; and the lower layers contain the processes roles, which are responsible for developing the tasks.

The model proposed in this paper focuses on developing a planning mechanism to coordinate the agents found in the VO. We will first identify the roles that these agents can assume:

- *Processor role:* Responsible for carrying out the activities required for each specific task. For this reason, the agent will specialize in a specific activity according to the type of tasks the system must resolve.
- *Planner role:* Designs the overall plan to be implemented by the organization. Sets the number of processor agents and distributes tasks according to the role they play. Replans according to the size of the input queue or the inability to accomplish an activity with a plan.
- *Distributor role:* Distributes tasks according to their completion by the agents and checks that each task is being processed within the time limits established for the plan.
- *Coordinator*: Performs the general control of the system. Communicates with the THOMAS platform elements to carry out control actions (connect, disconnect, exceptions, etc.).
- *Manager role:* This agent manages all the information of the task and communicates to the user.

Fig. 1 illustrates the different agents of the system and the interactions among them. The task list, which stores the activities to carry out in the multi-agent system, is shown in the upper corner; the agents and the interconnections can be seen in the center of the image.

The service layer is responsible for carrying out the functionality of the different agents. This ensures a separation between the functionality and the business logic. The behavior of the agents is independent of the agent itself since the functionality depends on the roles that the agent has at a particular moment. In order to achieve this separation, the role distributor assigns web services that integrate the behavior of CBP-BDI and reactive models associated with each role. The reactive models are based on the definition of dynamic rules that are analyzed and interpreted by the BRMS (Business Rule Management System). There two different services associated with the roles: planner and reactive. The planner service incorporates tasks from the CBP-BDI model assigned to the planner role, while the reactive model incorporates a business rules reasoning engine that interprets the business logic of the system.

The service layer includes 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 (Foundation for Intelligent Physical Agents-Agent Communication Language) and SOAP (Simple Object Access Protocol) protocols (Bauer & Huget, 2003).

One of the major problems in the development of an architecture based on a multi-agent system is that there are currently no clear standards or well developed methodologies for defining the steps of analysis and design that need to be taken. There are at present a number of methodologies: Gaia (Wooldridge, 2000), AUML (Agent UML) (Corchado & Laza, 2003), INGENIAS (Pavón, Gómez, Fernández, & Valencia, 2007), TROPOS (Giorginia, Mylopoulos, & Sebastiani, 2005), MESSAGE (EURESCOM, 2001). The main approach of agent-oriented methodologies is centered on the individual actions of the agents, many of which have been developed from extensions of object-oriented methodologies or from knowledge engineering, while others have started from totally independent developments. The majority of these methodologies define the phases of analysis and design, whereas only some (like Tropos, Prometheus, MaSE, MASSIVE) also detail the implementation phase. They assume that the agents have common objectives, are benevolent, and cooperate with each other to reach these common objectives. The social rules are not specifically defined, and the social structure, which is presented by the system, emerges from the interactions of the agents, yet remains undefined as well, both in the analysis and in the design of the actual methodology. For this reason, these types of methodologies are not useful by themselves for the development of open multiagent systems, and they only allow the development of closed systems in which the participation of external agents is not admitted, is normally nonreliable and uncooperative, and includes self-interested behaviours (Argente, 2005).

The main approach of an organization-oriented methodology is centered on its own organization of the system, and takes its objectives, structures and social norms into account. These methodologies primarily detail the analysis phase and in some cases the design phase. However, it is necessary to have an agent-oriented

Please cite this article in press as: Zato, C., et al. Model for assigning roles automatically in egovernment virtual organizations. *Expert Systems with Applications* (2012), doi:10.1016/j.eswa.2012.01.185

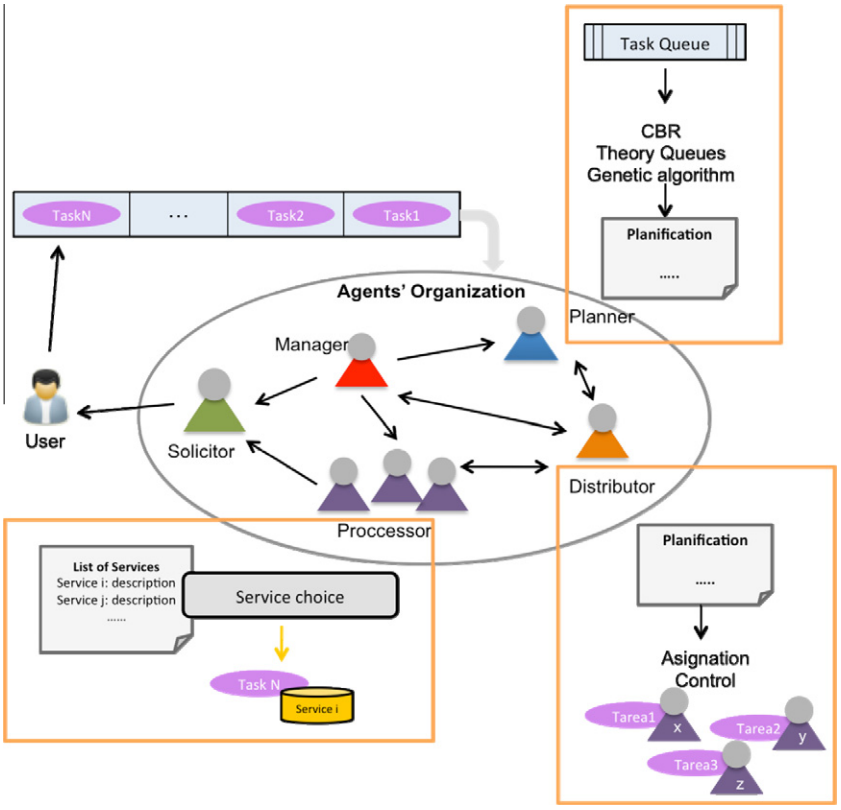

Fig. 1. Outline of proposed model.

methodology to complete the internal aspects of the agents and the implementation of the system, even though in the majority of the cases there is no explanation as to how this support is actually done. In addition, some of the proposals are merely extensions of agent-oriented methodologies, although many are independent developments that apply organizational theories and surroundings as the bases.

When the concept of organization is identified as the central point of the methodology, the social structure will be specifically defined for this purpose, indicating the objectives, roles, hierarchies, groups, interactions and topology of the system. Assuming the agents are heterogeneous, the participation of external and internal agents is allowed. Furthermore, the social norms define not only the mechanisms for including external agents within the society, but also the control mechanisms for the behaviour of the agents according to the restrictions imposed by the system. For these reasons, these methodologies are the most suitable for modelling both open and closed multiagent systems. Nevertheless, they are still in an incipient phase of development and lack some formalisms and appropriate methodological guidelines for carrying out the analysis, design and complete implementation of a multiagent system and, especially, of an open system (Argente, Julián, Valero, & Botti, 2005).

The present study uses GORMAS methodology because it covers all the specific needs that arise with virtual organizations, having been created precisely for this purpose. In particular, the methodological guide offers an iterative procedure that makes it possible to: (i) specify the mission of the system; (ii) analyze the required tasks and processes, on the basis of services and products; (iii) determine the dimensions of the organization (departmentalization, specialization, centralization, coordination and normalization); (iv) select the suitable structure according to the specified dimensions; (v) identify the processes of information and decision; (vi) specify the open characteristics of the system (functionality to publish; control of external agents); (vii) determine the mechanisms (rules) for controlling agent behaviours; and (viii) specify the reward system to promote the behaviours that most interesting to the organization (Valero, Argente, Giret, Julian, & Botti, 2005).

The following sections detail the two most innovative elements in the system: the CBP-BDI task planning role that specializes in coordinating tasks; and a dynamic distribution mechanism for tasks and roles based on the creation of rules.

4.1. Role planner

The role planner is responsible for carrying out the planning as the tasks are received. This agent is in charge of receiving the task list, establishing the number of agents needed to accomplish the activities, and putting the tasks in order so that any delays have the least possible impact on the overall plans. In Fig. 2, we can see the information stored in the task list. The information for each task is:

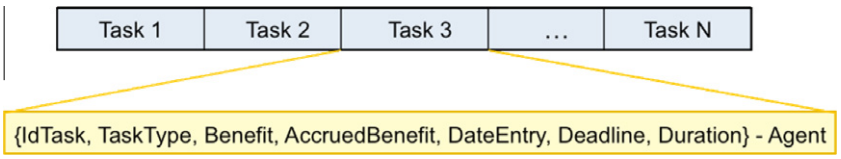

Fig. 2. Task list information.

Please cite this article in press as: Zato, C., et al. Model for assigning roles automatically in egovernment virtual organizations. *Expert Systems with Applications* (2012), doi:10.1016/j.eswa.2012.01.185

- IdTask: Identifies the global task in the system with a unique ID.
- TypeTask: Each task requires a set of activities that can be performed sequentially or in parallel. These activities can be considered as subtasks of the global task. This field establishes the current state of the global task and, consequently, the next activity or step that must be performed. From this point forward, the word task will be employed to refer to any activity performed to resolve the final task, bearing in mind that each task is defined by IdTask and TypeTask.
- Profit: Total profit gained from executing a task for a specific case. This field is important when determining the number of agents needed to minimize losses from not performing certain tasks..
- Accumulated profit: Total accumulated profit from previously completed subtasks that will be lost if the current task is not completed on time.
- Arrival time: This date establishes the arrival time of the task to the system. This field is important for determining the arrival rate of specific activities for a given date.
- Deadline: Maximum amount of time allowed for resolving the case. This value is preset for each type of task.
- Duration: Total time used in finalizing the task.
- Agent: The agent responsible for carrying out a task.

Based on the list of cases that need to be developed, the role planning agent selects the number of agents needed to support demand while minimizing costs, and distributes tasks among the agents. The tasks are distributed in so as to minimize any replanning required to complete the tasks in the previously determined time. This process is summarized in the Fig. 3, which displays a task list received by the planner agent, who then determines the most suitable number of processor agents and distributes the tasks.

If there are not enough agents to finalize the tasks, the system puts the cases in the best order to minimize losses. The system should always work under stable conditions, which is to say, those in which the utilization rate is less than 1.

The replanning process can be carried out according to different situations:

- New task: The system replans when it receives a new task. This mechanism would not be very useful if the arrival rate were high because the system would spend too much time replanning.
- Periodically: The system replans every specific period of time. If the system were to follow this procedure, it would be necessary to establish the period of time based on the arrival rate. However, as the arrival rate is not constant, this procedure could be ineffective. For example, it may be necessary to replan with the addition of only one task, while twenty new tasks are waiting to be reviewed.
- Accumulated number of unassigned tasks: Instead of replanning each time a new task arrives, the system replans only after there are a specific number of tasks waiting to be assigned to an agent.
- Inability to schedule: the system initiates the replanning process when delays by the processing agents have made it impossible to complete the most recent plans.

The first two options can be considered as simple systems with few tasks. In order to build a system that can adapt to different scenarios, the proposed system replans according to the number of tasks waiting to be assigned, and whether the previous plan can be successfully completed. The system establishes a threshold of 5%, based on the number tasks in the previous queue, before initiating a replanning process. It also creates a new distribution upon receiving an urgent task whose duration is similar to the maximum resolution term.

4.2. Planification model

The planning model explained in this section is associated with the planner role. The role planner agent has the capability to learn from previous cases. The agents adopt the CBP (Case-Based Planning) model, which is a special type of CBR (Kolodner, 1993).

While CBP is derived from CBR, it is specially designed to generate plans (sequences of actions) (Castro, Navarro, Sánchez, & Zurita, 2009; Corchado, Bajo, De Paz, & Tapia, 2008). In CBP, the proposed solution for solving a given problem is a plan. This solution is generated by taking into account the plans applied for solving similar problems in the past. The problems and their corresponding plans are stored in a memory of plans. The reasoning mechanism generates plans using past experiences and planning strategies, which is how the concept of Case-Based Planning is obtained (Glez-Bedia & Corchado, 2002). The problem description (initial state) and solution (situation when final state is achieved) are represented as beliefs, the final state as a goal (or set of goals), and the sequences of actions as plans. The CBP cycle is implemented through goals and plans. When the goal corresponding to one of the stages is activated, different plans (algorithms) can be executed concurrently to achieve the goal or objective. Each plan can activate new sub-goals and, consequently, cause the execution of new plans. In practice, what is stored is not only a specific problem with a specific solution, but also additional information about how the plans have been derived. As with case-based reasoning, the case representation, the organization of the memory of plans, and the algorithms used in every stage of the case-based planning cycle are essential in defining an efficient planner.

To carry out the planning process the agent follows the CBR-BDI planning model. The first point in the definition of a CBR-BDI model is the definition of case (1).

$$C = \{t_i / t_i = (ids, idt, b, B, f, p, d, a), \quad i = 1 \ldots n\} \tag{1}$$

where t_i presents the task i, ids is the identifier of the task type, idt the process, b benefit, B the accrued benefit, f the entry, p the term, d the duration, and a the ID of the agent that performed the task. The application of the different stages of reasoning is performed as follows:

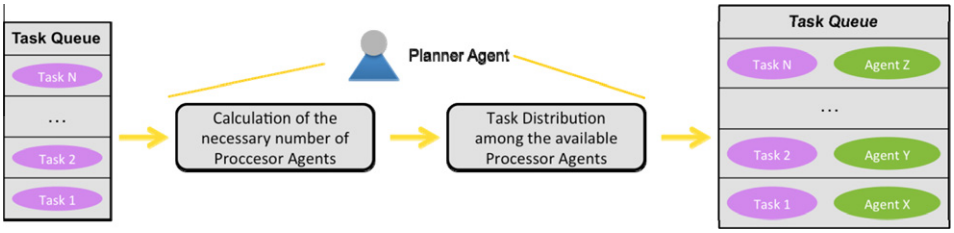

Fig. 3. Planning process.

Please cite this article in press as: Zato, C., et al. Model for assigning roles automatically in egovernment virtual organizations. *Expert Systems with Applications* (2012), doi:10.1016/j.eswa.2012.01.185

```
Input: C_r, c_{n+1}, N, eMax, cost
Output: number, TaskList, Average

Error ← ∅;
Average ← ∅;
TaskList ← retrieveTask(c_{n+1});
foreach x ∈ TaskList do
    e ← calculateError(x, C_r, N_x);
    if e > eMax then
        SimilarTasks ← retrieveSimilarTasks(x, eMax);
        Average ← calculateAverage(c_r, SimilarTasks);
    end
    else
        Average ← calculateAverage(C_r);
    end
    Error ← Error ∪ {e};
    Average ← Average ∪ {Average};
end
[number, TaskList] ← calculateAgents(C_r, Average, cost);
```

Fig. 4. Definition of the number of total services.

Upon receiving a new case c_{n+1} composed of the elements indicated in (1), the planning agent creates a new plan by applying the different stages of the reasoning cycle as described in the following sections:

In retrieve phase the system retrieves the most similar cases to the current case c_{n+1}. The most similar cases are those that contain the greatest number of tasks most similar to those in the current queue. In order to improve the search process, information about the number of tasks for each type of activity is added to the information in the memory of plans. The search of the stored tasks is limited to a predefined period of time, which can vary according to the type of services that have been given to the organization. The number of recovered cases is defined in advance in order to subsequently carry out the reuse phase. The set of retrieved cases is defined as $C_r \subseteq C$.

During the revise phase, the system uses the information retrieved from the memory of cases C_r in order to generate the plan associated with the new case c_{n+1}. The retrieved information is adapted by means of the queuing theory and genetic algorithms. The retrieved information is used, first of all, to determine the arrival rate and the service rate for each case. The queuing theory then determines the number of services needed for a case. The information obtained from the retrieved cases is used to calculate the average service time. Statistic sampling is used to calculate the maximum error for the value obtained with a specified confidence level. If after calculating the error the specified threshold is exceeded, the system will select the size of the minimum sample to reduce the error down to the indicated value so that the threshold is no longer exceeded. Fig. 4 shows the algorithm used to calculate the total number of required services.

The calculateServices function is carried out by the algorithm shown in Fig. 5; the process is described in detail in Section 4.2.2.

The algorithm used during the reuse phase is described in Fig. 6. The system distributes tasks among the agents according to a neural network. It already has the list of tasks to organize, the length of each task based on the agents, and the maximum time allowed to complete the process. The exit matrices for the WeightEstimateHiddenLayer and WeightEstimateOutputLayer algorithms contain the assignments and execution order of the tasks for each of the agents. The estimation process followed by the neural network is explained in greater detail in Section 4.2.3. The learning process uses the common retro-propagation algorithm.

The **revise** phase is automatically initiated as the agents finish their tasks. The agent updates the duration of the tasks as they

```
Input: C_r, Average, cost
Output: number, Tasks

SortedTasks ← sortProfitTask();
f ← 0;
Tasks ← ∅;
for i=1 to #SortedTasks do
    L_q ← QueuingElements(C_r, Medio[i], s=1);
    W_q ← calculateAverageWait(L_q, λ);
    L ← SystemElements(L_q, λ, Average[i]);
    duration ← calculateAverageDuration(C_r, SortedTasks[i], eMax);
    profit ← calculateAverageProfit(C_r, SortedTasks[i], eMax);
    [f_b, services] ← calculateProfit(L, Averaga[i], duration, profit);
    p ← updateUtilizationRate(λ, Average[i]);
    if ρ > 1 then endFor;
    f ← f+f_b;
    Tasks ← Tasks ∪ SortedTasks[i];
    number ← number + services;
end
```

Fig. 5. Calculation of the number of services.

```
Input: Tasks, Duration, maximumTime
Output: Tasks, WeightEstimateHiddenLayer,
        WeightEstimateOutputLayer

iteration ← 0;
while iteration ≤ Max && !constantError do
    [WeightEstimateHiddenLayer, WeightEstimateOutputLayer,
    OutputHiddenLayor, OutputOutputLayer, output] ← compute(Tasks,
    Duration, maximumTime, WeightHiddenLayer, WeightOutputLayer);
    [WeightHiddenLayer, WeightOutputLayer] ←
    learnBackPropagation(OutputHiddenLayor, OutputOutputLayer,
    Tasks, output);
end
```

Fig. 6. Task assignment.

are completed and then replans if it receives a message from the processor agent regarding the impossibility of completing a plan under the existing temporal restrictions.

The **retain** phase stores the plan at the end of the business day. The new memory of cases C' is defined as follows: $C' = C \cup c_{n+1}$ Any plans that originated prior to a specific date are removed in order to limit the size of the memory.

4.2.1. Calculation of maximum error

In order to estimate the number of processor agents needed and to assign tasks, it is first necessary to estimate the average duration for each kind of task in general and for each of the agents in particular. The process of calculating the duration of tasks is done by first determining thelevel of maximum error allowed for a given confidence level.To this end, the system establishes the level of confidence and calculates the level of error obtained from a sample size. Eq. (2) defines the average interval of a given sample and population.

$$\hat{\mu} = \bar{x} \pm k \frac{s}{\sqrt{n-1}} \sqrt{\frac{N-n}{N-1}} \tag{2}$$

where $\bar{x}$ represents the average, k is the value Z of a $N(0,1)$ that belongs to a predetermined level of confidence, N is the size of the population, n is the sample size, and s is the deviation.

Eq. (2) is simplified if $N \gg n$ by approximately 100 times. Under this condition, the equation can be reduced to the following expression (3).

$$\hat{\mu} = \bar{x} \pm k \frac{s}{\sqrt{n_\infty}} = \bar{x} \pm e \tag{3}$$

where n_∞ represents that $N \gg n$ by approximately 100 times.

Therefore, we can obtain the expression (4) that determines the sample size used to establish a maximum error.

Please cite this article in press as: Zato, C., et al. Model for assigning roles automatically in egovernment virtual organizations. *Expert Systems with Applications* (2012), doi:10.1016/j.eswa.2012.01.185

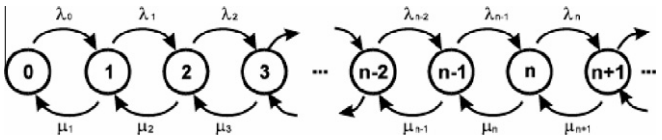

Fig. 7. Birth–death process.

$$n_\infty = \frac{k^2 S_c^2}{e^2} \tag{4}$$

S_c represents the quasivariance; the others variables have been previously indicated.

Finally, the sample size is defined by Eq. (5) in the first stages of the system when the condition $N \gg n$ is not satisfied.

$$n = \frac{1}{\frac{1}{n_\infty} + \frac{1}{N}} \tag{5}$$

The equations presented in this section establish the number of elements that must be retrieved in order to limit any obtained errors. This process can limit access to the memory and remove old cases that have not already been accessed.

4.2.2. Dynamic planning roles

The number of agents that should be available in the system is estimated dynamically. The intention is for the number of agents to meet the demand and ensure that the system utilization factor ρ is less than 1. The queuing theory is used to calculate the prediction. The queuing theory originated from a study of telephony in Denmark in 1909 (Erlang, 1909). The study involved the provision of services to clients with an unknown set of demands. Queuing theory was expressed as a birth-death process, which defines the relationship between the arrival of tasks and their service so that the problem balances out in such a way that all tasks can be addressed. Fig. 7 illustrates this problem.

The likelihood of the system belonging to state k can be defined according Fig. 7, where state k means that there are k tasks in the system queue.

If there are no tasks in the system queue, the relationship between the input to state 0 and the output to this state can be defined as:

$$\mu_1 P_1 = \lambda_0 P_0 \tag{6}$$

$$P_1 = \frac{\lambda_0}{\mu_1} P_0 \tag{7}$$

Using the previous expression (6) we can assume that the probability of finding zero tasks in the system queue (P_0) multiplied by the arrival rate must be equal to the probability of finding one task in the system queue that will be addressed immediately.

The following expression can be defined in the state n-1

$$\lambda_{n-2} P_{n-2} + \mu_n P_n = \lambda_{n-1} P_{n-1} + \mu_{n-1} P_{n-1} \tag{8}$$

$$\mu_n P_n = \lambda_{n-1} P_{n-1} + \mu_{n-1} P_{n-1} - \lambda_{n-2} P_{n-2} \tag{9}$$

$$P_n = (\lambda_{n-1} P_{n-1} + \mu_{n-1} P_{n-1} - \lambda_{n-2} P_{n-2})/\mu_n \tag{10}$$

Using (10) and replacing P_{n-1} we have

$$P_n = \frac{\lambda_0 \cdots \lambda_{n-1}}{\mu_1 \cdots \mu_n} P_0 = c_n P_0 \tag{11}$$

The problem of planning multiple tasks can be simplified to a case of planning a single task for each type of task. Thus, a plan is performed independently for each task so that the average waiting time and average queue length can be calculated independently. The average waiting time and the overall average length is simplified to calculating the average values for each of the tasks. In the case of the M/G/s model where $s = 1, 2, 3, \ldots$ is the number of agents, the arrival rate $\lambda_n = \lambda =$ etc, the service rate for when there are n processes is defined by the following Eq. (12) (Martín, 2003):

$$\mu_n = \begin{cases} N\mu & n = 1, 2, \ldots, s-1 \\ s\mu & n \geq s \end{cases} \tag{12}$$

where μ represents the average service rate for s available agents. This value depends on both the agents and the machine found.

Assuming that the system is in a stable condition (i.e., it meets the utilization factor $\rho = (\lambda/\mu s) < 1$), the probability that n tasks exists (P_n) in the system is given by Eq. (13) (Martín, 2003)

$$P_n = c_n P_0 = \begin{cases} \frac{\lambda^n}{n!\mu^n} P_0 & n = 0, 1, \ldots s-1 \\ \frac{\lambda^s}{s!\mu^s} \left(\frac{\lambda}{s\mu}\right)^{n-s} P_0 & n \geq s \end{cases} \tag{13}$$

where

$$P_0 = \frac{1}{\sum_{n=0}^{s-1} \frac{\lambda^n}{n!} + \frac{\lambda^s}{s!\mu} \frac{1}{1-\frac{\lambda}{s\mu}}}$$

$$c_n \begin{cases} \frac{\lambda^n}{n!\mu^n} & n = 1, 2, \ldots s-1 \\ \frac{\lambda^s}{n!\mu^s} \left(\frac{\lambda}{s\mu}\right)^{n-s} & n \geq s \end{cases} \tag{14}$$

Having defined the probability that n tasks exist in the system, it is possible to define the number of tasks L_q in the system queue and the average waiting time W_q of tasks at the end of the system (15) (Martín, 2003):

$$L_q = \sum_{n=s}^{\infty} (n-s) P_n = P_0 \frac{\lambda^s}{s!\mu^s} \frac{\lambda}{s\mu} \frac{1}{(1-\frac{\lambda}{s\mu})^2}, \quad W_q = \frac{L_q}{\lambda} \tag{15}$$

To determine the optimal number of agents we make an estimate that minimizes the cost function, which depends on both the number of agents used and the waiting time in the queue. The function is defined in a particular way for each service depending on the actual costs of each agent in the system. The following profit function is provided (16).

$$f(L, P_0, \ldots, P_{s-1}, \mu', \bar{p}, \bar{b}) = f_b(L, \mu', \bar{p}\bar{b}) - k \cdot s \tag{16}$$

$$F_b(L, \mu', \bar{p}\bar{b}) = \begin{cases} (\bar{p}/u')\bar{b} \cdot s \cdot (1-\rho) & \text{si} L \cdot u' > \bar{p} \cdot s \cdot (1-\rho) \\ L\bar{b} & \text{si} L \cdot u' \leq \bar{p} \cdot s \cdot (1-\rho) \end{cases} \tag{17}$$

where k is a constant associated with the cost of having an agent working, $\bar{b}$ the average benefit of performing the task, μ' is the average time to complete the task, obtained from the service rate, $\bar{p}$ the average time to execute a task. If we exceed the conditions of stability, f_b is counted only up to the utilization factor 1. The utilization factor ρ varies according to the new services added to the queue until it reaches the utilization factor of 1.

Following the cost function given in (7), we introduce the global cost function (18) that takes into account the implementation of the various services.

$$L_q = \sum_{n=s}^{\infty} (n-s) P_n = P_0 \frac{\lambda^s}{s!\mu^s} \frac{\lambda}{s\mu} \frac{1}{\left(1-\frac{\lambda}{s\mu}\right)^2} \quad W_q = \frac{L_q}{\lambda} \tag{18}$$

where f_i is calculated from Eq. (7). Because the stability conditions may not always be given, it is necessary to calculate the terms in order of benefit depending on the type of the task so that when you reach the utilization factor of 100% we end calculating the summation terms. Once the optimization function defined in (9) the maximum value is calculated iteratively starting with number of agents equal to 1, the fixed value is the first local maximum that corresponds to the global maximum.

Please cite this article in press as: Zato, C., et al. Model for assigning roles automatically in egovernment virtual organizations. *Expert Systems with Applications* (2012), doi:10.1016/j.eswa.2012.01.185

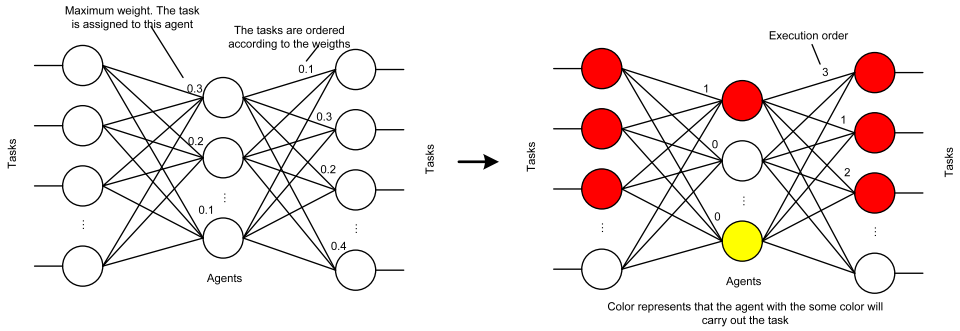

Fig. 8. Structure of the neural network that represents the task assignment and determines the order in which the tasks are executed.

4.2.3. Task assignment

The tasks are assigned by a multilayer perceptron that contains unsupervised learning. In this network the propagation rule and the error function used in the retro-propagation algorithm are modified in order to allow for unsupervised learning. The first point to consider with regards to the problem is the architecture of the network that will be used. The architecture is defined by the number of tasks and the number of available agents. In this way, the number of neurons at the entry and exit layers is defined by the number of tasks, while the number of the intermediate layers is established by the number of agents.

Once the network topology has been defined, the next step is to consider the problem needs and objectives. For the problem of assignment presented in this study, it is not only necessary to assign the tasks to the agents, but to consider the order in which the agents actually receive their assignments, given that the order of assignments is important to ensure the timely completion of the tasks. The assignment of the tasks and the order of assignment are determined according to a weighted value. The weight of the connection between the entry and the intermediate layers determines the assignment; the values are continuous during the learning phase, but they become binary during the estimating phase so that only the greatest value is 1 and the remainder are 0. The weight of the connexions between the neurons at the intermediate and the exit layers determine the order in which the tasks are executed. These values are continuous during the training phase, but during the estimating phase they become natural according to the position of the value given to the weight when it has exited the neuron of the intermediate layer. For those tasks with zero weight at the connection of the intermediate entry layer, the weight of the connection between the neuron for the intermediate and the exit layer is zero. Fig. 8 provides an example of the transformation of the continuous values to binary and whole numbers during the estimating phase. The first neuron at the entry layer is assigned to the first neuron from the intermediate layer because it has a greater weight value. At the exit layer, the order is determined by the maximum value of the weights for each of the neurons from the intermediate layer; this allows the second neuron at the exit layer to have a value of 1. The value of the weight of the last neuron at the exit layer is ignored since the last neuron at the entrance layer is associated with the second layer of the intermediate layer.

The first step in describing how the neural network operates will include a detailed description of the estimation phase in which weights are used to calculate the value of the neural network exit.

1. w^e_{ij} represents the weight element for the intermediate entrance layer that corresponds to the neuron i from the entrance layer, and j represents the neuron at the intermediate layer. The matrix is defined according to the the original weights w_{ij} from the network.

$$w^e_{ij} = \begin{cases} 1 & w_{ij} = \max_x w_{ix} \\ 0 & w_{ij} \neq \max_x w_{ix} \end{cases}$$

2. The exit of the neurons from the intermediate layer is represented by the following equation:

$$y_j = \sum_{w^e_{ij} x_i}$$

3. W^s_{jk} is the matrix for the weights from the intermediate exit layer, where j is the index of the entrance neuron and k is the exit neuron. The values for the matrix are defined as follows:

$$w^t_{jk} = \{w_{jk}/w_{ij} = 1, i = k\}$$
$$w^o_{jk} = \text{sort}(\{w_{jk}/w_{ij} = 1, i = k\})$$
$$w^s_{ij} = \{x/w^s_{jk} = w^o_{ix}\}$$

4. The exit of neuron k is defined according to the duration of the $x-1$ tasks preceding task k. The result is the sum of the duration of each of these tasks. Task i precedes task x if both tasks are associated with the same agent, and if the weight of task i is greater than that of task k. The duration depends as much on the agent as on the task itself.

$$z_k = \sum_{d_x}$$

5. The final error function is defined by the following expression in which M_k represents the maximum execution time for task k:

$$\frac{1}{2} \sum_{(M_k - z_k)}$$

Using the estimates for the exits, the learning phase can be carried out by applying the retro-propagation algorithm for the multilayer perceptron so that the weights adjust automatically, thus minimizing the definition of error.

5. Case study

The proposed system can be applied to different business environments and organizations that require task planning. However, this study was primarily motivated by the need for a platform that could create an effective distribution of work and fall in line with the new business goals generated by a recently installed e-Administration system. The result of this particular innovation within the field of Public Administration is to establish a new work mechanism that requires changes in the organization.

The case study presents a society that focuses on processing cases of Public Administration that have been received by telematics means. This society of agents was designed using the GORMAS design methodology, which uses various stages of analysis, structural

Please cite this article in press as: Zato, C., et al. Model for assigning roles automatically in egovernment virtual organizations. *Expert Systems with Applications* (2012), doi:10.1016/j.eswa.2012.01.185

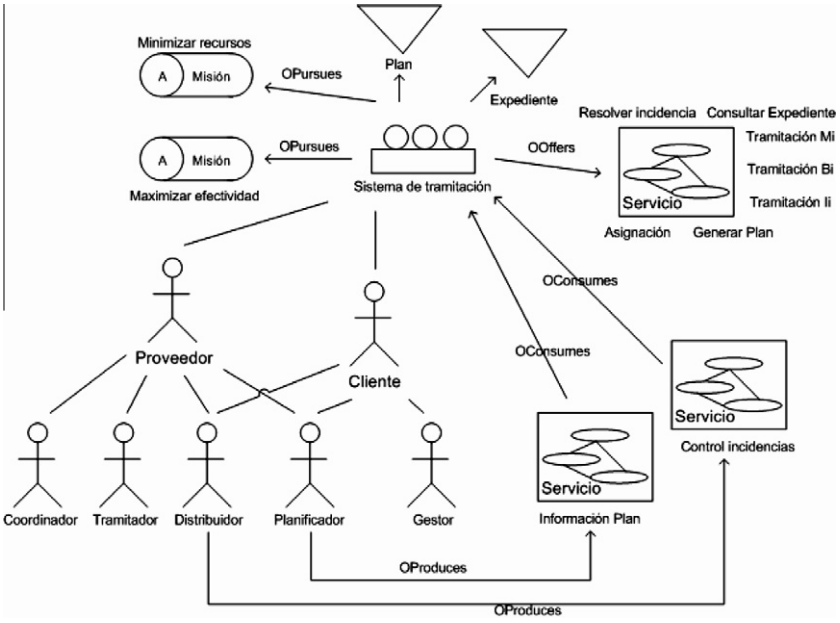

Fig. 9. Diagram of the organizational model. System mission.

design and dynamic design to specify the services offered by the organization, its structure, and the norms that dictate its behavior. The system was implemented and based on the previously mentioned THOMAS architecture (Carrascosa et al., 2009; Giret et al., 2009) and on the model proposed in Section 3. The system is meant to simulate the behavior of the Planner Agent, which ensures that the entire organization is provided with a global plan to successfully solve all cases.

The model proposed for this case study was adapted according to the particular specializations of the Processing Agents. These specializations are related to the three types of cases that will be taken into account: Scholarship Processor, Tax Processor, and Fine Processor. Each task in the entry queue is characterized by the case with which it is associated, and its current state or phase.

There are different services within the virtual organization, each of which can assume different roles. This way, when a Processer Agent receives a case, a service will be registered within the SF module of the THOMAS architecture. The service can be executed by the role that processes the case and will contain all the operations that the agent needs to carry out its task.

Following the methodological guidelines we can observe the results in Fig. 9 for one of the first tasks that must be performed. The figure instantiates the functional view (mission) of the organizational model, which includes the products and services offered by the system, the type of environment, the global objectives, interest groups and the information they consume.

6. Results

The program used in this study simulates the behavior of the Planner agent within a virtual organization. While the simulation is taking place, the actions are statistically analyzed for their performance and the system makes recommendations regarding the distribution of tasks.

Fig. 10 displays an image of the program in operation. The task queue shows the tasks waiting to be processed. Once the task was assigned, the Tax Processor was immediately delayed (as indicated by the red cross), requiring the Planner agent to replan (as indicated by the hourglass next to the Planner) since it has now become impossible to continue with the original plan.

With the aim of evaluating the proposed planning mechanism, and assuming the final goal is to process all the cases within the allotted time frame while yielding the greatest possible profit mar-

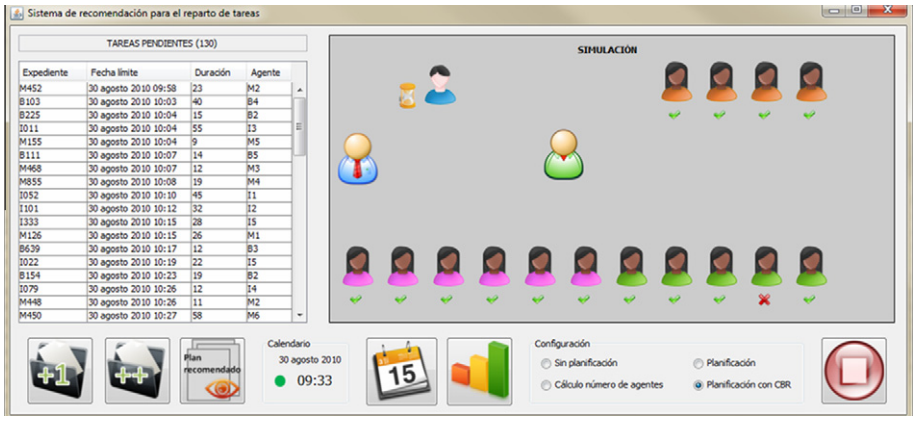

Fig. 10. System interface.

Please cite this article in press as: Zato, C., et al. Model for assigning roles automatically in egovernment virtual organizations. *Expert Systems with Applications* (2012), doi:10.1016/j.eswa.2012.01.185

Table 1
Configuration of the genetic algorithm.

Population: 20
Cross: multipoint
Mutation: alter the assignment order and alter the assignment
Elitism: 10%
Stop criteria: 500 iterations or no improvement in 4 generations

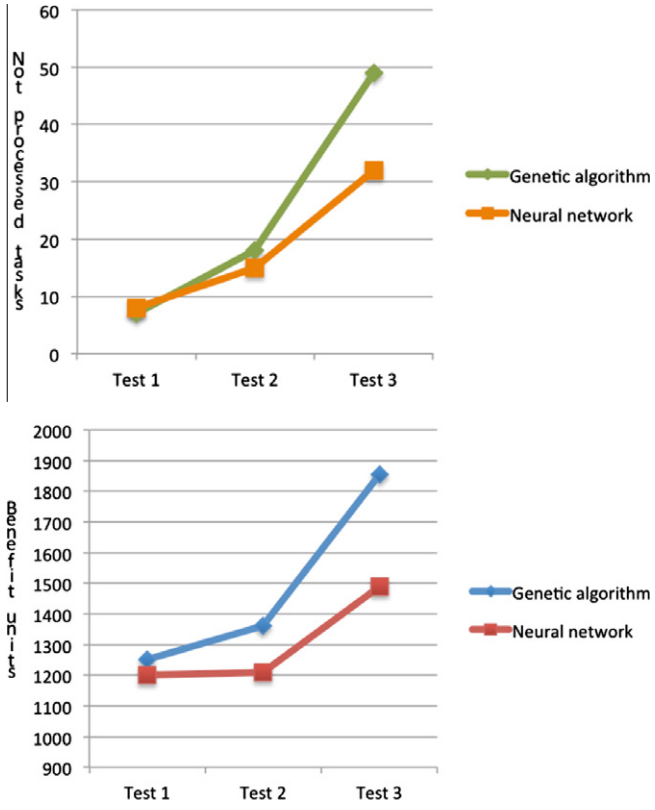

Fig. 11. Comparative results for task planning.

gin, the system was tested with three different case lists. As previously mentioned, prior versions of this planning model used a genetic algorithm to assign tasks. However, the following section will show that the results obtained from the neural network improve efficiency as well as profit and the number of cases successfully processed.

The genetic algorithm contained an aptitude function defined in the same way as the error function used for the neural network so that the results obtained would be comparable. The genetic algorithm used a multipoint cross-connection and mutation operators to alter both the order of task assignment to agents, and the assignment of the agents themselves. The mutation operators were selected randomly for each iteration; the random mutation of the least efficient chromosomes was also permitted to avoid falling into local minima. The configuration selected in the genetic algorithm is shown in Table 1.

Three sets of cases were designed:

- Test 1: 400 cases introduced within a simulated 32 hour workweek
- Test 2: 800 cases introduced within a period of 60 hours.
- Test 3: 1300 cases placed in the entry queue within a period of 70 hours

Previously, the system had a memory of 700 cases based on cases where the values had been altered by a poisson distribution

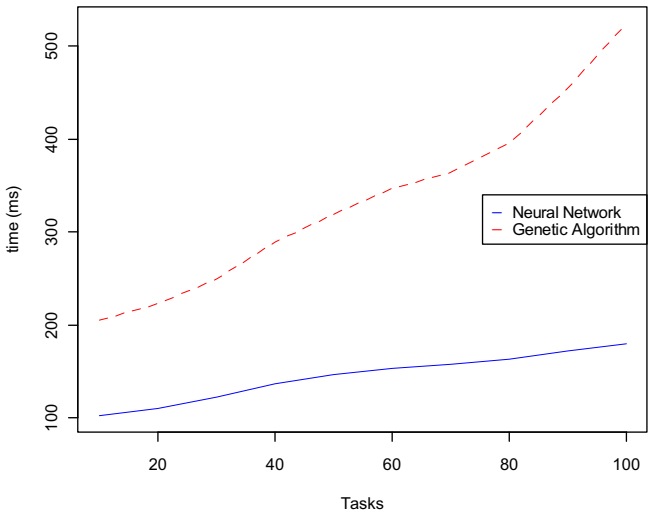

Fig. 12. Execution time in milliseconds for the tasks and estimate process.

according to the arrival rate established for each of the test cases. Fig. 11 shows the comparative data obtained from substituting the genetic algorithm with the neural network in the three test cases.

The top chart displays the comparison between the number of cases that were not processed within the stipulated period of time. For the first test, in which there were fewer cases, we can see that the network does not indicate an improvement in the results. However, as the number of cases increased and were introduced in a smaller period of time, the network reduced the number of unprocessed cases from 3.77% to a best case of 2.46%.

The bottom chart is even more significant because it reveals the lost profits. A point system is used to assign a value to the benefit field for each record. It takes various aspects into account, such as the type of record, the economic gain expected, previous work, the assigned agent, etc.

As with the test using lost cases, the first test does not demonstrate a significant improvement. However, the distance between losses is greater when the system employs a larger entry queue. The graph shows how the neural network in test 3 manages to reduce lost profits from 1845 units to 1490 units.

The comparison between the genetic algorithm and the neural network also included a comparison of execution times and the final efficiency rate for the solution obtained. To perform this comparison, we calculated the execution time for the network and the genetic algorithms for various numbers of tasks. The *x*-axis represents the number of tasks waiting in the queue prior to the planning, and the *y*-axis represents the time. The results can be seen in Fig. 12.

We can see that the execution time has a greater increase for the genetic algorithms while the increase in the neural network remains more proportional to the number of elements in the planning process, with a more linear movement than the genetic algorithm.

In the next step, we performed stress tests on the system, which randomly eliminated an agent prior to the planning process. Fig. 13 shows the number of tasks completed on time for each of the planners. We can see that the number of tasks completed on time is greater for the neural network than for the genetic algorithm.

For this study, the Mann-Whitney U-test was applied to the proportion of tasks finished on time. It was a non-parametric test in which it is not necessary to make assumptions on the data distribution, as with the t-test. The test determines two values: H_0 and H_1. H_0 shows whether the data in both groups presents the same distribution, whereas H_1 determines whether there is a

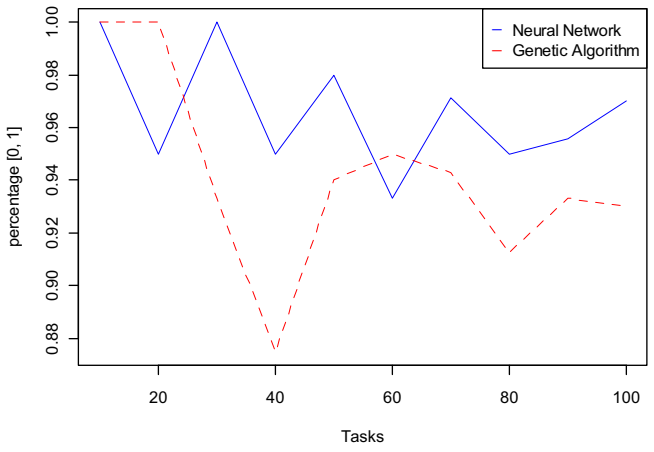

Fig. 13. Percentage of tasks completed on time for different numbers of tasks.

difference in the distribution of the error distance data. The result obtained was 0.0473, which implies that the difference is significant and, consequently, that the proportion of unfinished tasks is the result of a different distribution.

7. Conclusions and future work

This article has presented a virtual organization model with a task planning mechanism that can make recommendations when distributing tasks in a dynamic environment. Virtual organizations were proposed as the mechanism for a business model and included all of the characteristics that compose a real life scenario. Because of their many advantages, virtual organizations are ideal for simulating the behavior of a business entity in which planning and coordination are the keys to success.

The use of the neural network as a mechanism for exploring the search area and achieving optimal productivity was effective within the proposed problem, which is characterized by the complexity that results from the significant number of options and combinations available. The use of the queuing theory allowed us to establish the number of agents needed to optimize resources. Both methods were satisfactorily incorporated into the CBR reasoning cycle, specifically in the reuse phase. This demonstrates once again the facility in extending the CBR-BDI architecture to incorporate different techniques in its functionality.

The proposed model was developed using a real-life scenario, with the e-Administration providing a detailed explanation of the required steps for adapting the model, identifying the roles, and other details regarding the characterization of the different tasks.

Finally, one of the principal achievements of this work is in validating the proposed planning model by means of an implemented simulation. The results obtained confirm that the planning process provided notable improvement and, in refining the cases used in the CBR reasoning cycle, also achieved a higher level of learning and adaption within a dynamic environment.

Future work includes:

- To explore other heuristic and metaheuristic methods to test different formulas for assigning tasks. To establish a value of effectiveness for each method and provide the user with recommendations for different plans according to their success rate.
- To include the developed model within a complete e-Adminsitration system. This system could provide a more complete control of personnel, cases and resources, while allowing for efficient distribution within the organization.
- To design and implement the complete organization, including a model of all of the required policies and services.
- Future work would propose a revision in the neural network to address the anomalies in the functionality that were detected in the present study, specifically the problem of giving equal weight to the delays and progress. This problem is important if the utilization factor were particularly low, although if the factor were close to 1 it would not make a significant difference. The queuing theory contributes towards ensuring that the utilization factor remains high. To solve this problem, it would be necessary to modify the defined error function so as to adjust the importance of delays and thus refine the retropropagation algorithm being used. The only adjustment that is currently being made to avoid this problem is with the error arising from delayed tasks.

Acknowledgements

Special thanks to the Institute of Cancer of Salamanca for the information and technology provided. This work was supported by the Spanish Ministry of Science TIN 2009-13839-C03-03 Project.

References

Andres, C., Miralles, C., & Pastor, R. (2008). Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. *European Journal of Operational Research, 187*(3). 1212-122.

Ardissono, L., Petrone, G., & Segnan, M. (2004). A conversational approach to the interaction with Web Services. *Computational Intelligence, 20*, 693–709.

Argente, E. (2005). A proposal for an organizational-oriented MAS methodology. In *Proceedings of the autonomous agents and multi agent systems AAMAS' 05* (pp. 1370). ACM Press.

Argente, E., Julián, V., Valero, S., & Botti, V. (2005). Towards an organizational MAS methodology. In *Proceedings of the CCIA'05. Recent advances in artificial intelligence research and development. Frontiers in artificial intelligence and application* (pp. 397–404). IOS Press.

Bajo, J., Corchado, J. J., Botti, V., & Ossowski, S. (2009). Practical Applications of Agents and MAS: Methods∗ Techniques and Tools for Open MAS. *Journal of Physical Agents, 3*(2).

Bauer, B., & Huget, M. P. (2003). FIPA modeling: Agent class diagrams. Working draft, Foundation for Intelligent Physical Agents. <www.auml.org>.

Binato, S., Hery, W., Loewenstern, D., & Resende, M. G. (2000). *A GRASP for job scheduling*. Technical Report No. 00.6.1 AT&T Labs Research.

Boella, G., Hulstijn, J., & van der Torre, L. (2005). Virtual organizations as normative multiagent systems. In *Proceedings of the 38th Hawaii international conference on system sciences* (pp. 1–10).

Camarinha-Matos, L. M., & Afsarmanesh, H. (2007). A framework for virtual organization creation in a breeding environment. *Annual Reviews in Control., 31*, 119–135.

Carrascosa, C., Giret, A., Julian, V., Rebollo, M, Argente, E., & Botti, V. (2009). Service oriented MAS: An open architecture (short paper). In *Proceedings of 8th international conference on autonomous agents and multiagent systems (AAMAS 2009), Budapest, Hungary* (pp. 1291–1292).

Castro, J. L., Navarro, M., Sánchez, J. M., & Zurita, J. M. (2009). Loss and gain functions for CBR retrieval. *Information Science, 179*(11), 1738–1750.

Chambers, J., & Barnes, W. (1996). *Taboo search for the flexible-routing job shop problem*. Technical report TAY 2.124, Department of Computer Sciences, University of Texas, USA.

Corchado, J. M., Bajo, J., De Paz, Y., & Tapia, D. I. (2008). Intelligent environment for monitoring alzheimer patients, agent technology for health care. *Decision Support Systems, 44*(2), 382–396.

Corchado, J. M., & Laza, R. (2003). Constructing deliberative agents with case-based reasoning technology. *International Journal of Intelligent Systems, 18*(12), 1227–1241.

David, N., Marietto, M. B., Sichman, J. S., & Coelho, H. (2004). The structure and logic of interdisciplinary. Research in agent-based social simulation. *Journal of Artificial Societies and Social Simulation, 7*(3).

Durfee, E. H., Lesser, V. R., & Corkill, D. D. (1989). Trends in cooperative distributed problem solving. *IEEE Transactions on Knowledge and Data Engineering, 1*(1), 63–83.

Epstein, J. M., & Axtell, R. L. (1996). *Growing artificial societies: Social science from the bottom up*. MIT Press.

Erlang, A. K. (1909). The theory of probabilities and telephone conversations. *Nyt Tidsskrift for Matematik B, 20*, 131–137.

Esteva, M., Rodríguez, J., Sierra, C., Garcia, P., & Arcos, J. (2001). On the formal specific ations of electronic institutions. *Agent-mediated Electronic commerce*, 126–147.

EURESCOM (2001). *MESSAGE: Methodology for engineering systems of software agents*. Technical report P907-TI1, EURESCOM.

Fenga, D. Z., & Yamashiro, M. (2006). A pragmatic approach for optimal selection of plant-specific process plans in a virtual enterprise. *Journal of Materials Processing Technology, 173*(2,10), 194–200.

Giorginia, P., Mylopoulos, J., & Sebastiani, R. (2005). Goal-oriented requirements analysis and reasoning in the Tropos term methodology. *Engineering Applications of Artificial Intelligence, 18*(2), 159–171.

Giret, A, Julian, V. Rebollo, M., Argente, E., Carrascosa, C., & Botti, V. (2009). An open architecture for service-oriented virtual organizations. In *Seventh international workshop on programming multi-agent systems. PROMAS 2009* (pp. 23–33).

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

Jennings, N., & Wooldridge, M. (1998). *Applications of Intelligent Agents*. Queen Mary & Westfield College: University of London.

Kabadi, S., & Punnen, A. (2008). A strongly polynomial simplex method for the linear fractional assignment problem. *Operations Research Letters, 36*(4), 402–407.

Kim, K. H., & Park, Y. (2004). A crane scheduling method for port container terminals. *European Journal of Operational Research, 156*(3), 752–768.

Kolodner, J. (1993). *Case-based reasoning.* San Francisco: Morgan Kaufmann.

Kolonko, M. (2009). Some new results on simulated annealing applied to the job shop scheduling problem. *European Journal of Operational Research, 113*(1), 123–136.

Lee, D. H., Wang, H. Q., & Miao, L. X. (2008). Quay crane scheduling with non-interference constraints in port container terminals. *Transportation Research Part E, 44*, 124–135.

Liu, X. (2007). A multi-agent-based service-oriented architecture for inter-enterprise cooperation system. In *Proceedings of the second international conference on digital telecommunications (ICDT'07)*. Washington, DC: IEEE Computer Society.

Markus, G., Ingo, S., Josef, F., & Helmut, B. (2010). The formation of virtual organizations by means of electronic institutions in a 3D e-Tourism environment. *Information Sciences, 180*, 3157–3169.

Martín, Q. (2003). *Investigación Operativa.* Prentice-Hall.

Oliva, E., Natali, A., Ricci, A., & Viroli, M. (2008). An adaptation logic framework for java-based component systems. *Journal of Universal Computer Science, 14*(13), 2158–2181.

Pacheco, J. A., & Casado, S. (2003). Estudio Comparativo de Diferentes Metaheurísticas para la Resolución del Labor Scheduling Problem. *Estudios de Economia Aplicada, 21*(3), 537–557.

Pavón, J., Gómez, J., Fernández, A., & Valencia, J. (2007). Development of intelligent multi-sensor surveillance systems with agents. *Journal of Robotics and Autonomous Systems, 55*(12), 892–903.

Porto, S., Kitajima, J. P., & Ribeiro, C. (1996). Performance evaluation of a parallel tabu search task scheduling algorithm. *Parallel Computing, 26*(1), 73–90.

Rigole, P., Holvoet, T., & Berbers, Y. (2002). Using Jini to integrate home automation in a distributed software-system. In *Fourth international workshop on distributed communities on the web, April 03–05, 2002* (Vol. 2468(2), pp. 91–304).

Rocha, A., & Oliveira, E. (1999). An electronic market architecture for the formation of virtual enterprises. In *Proceedings of the IFIP TC5 WG5.3/PRODNET working conference on infrastructures for virtual enterprises: Networking industrial enterprises, 1999*.

Rodríguez, P. (2003). *Discusión y Análisis de la metaheurística SN*. Depto. Investigación Operativa, InCo, FI, UdelaR. Reporte Tecnico 03-02.

Schelling, T. (1978). *Micromotives and macrobehavior*. New York: W.W. Norton.

Seda, M. (2007). Mathematical models of flow shop and job scheduling problems. *World Academy of Science, Engineering and Technology, 31*, 122–127.

Shafiq, M. O., Ding, Y., & Fensel, D. (2006). Bridging multi agent systems and web services: Towards interoperability between software agents and semantic web services. In *Proceedings of the 10th IEEE international enterprise distributed object computing conference (EDOC'06)* (pp. 85–96). Washington, DC: IEEE Computer Society.

Shi, C., Lu, J., & Zhang, G. (2005). An extended Kuhn–Tucker approach for linear bilevel programming. *Applied Mathematics and Computation, 162*(1), 51–63.

Shi, C., Lu, J., & Zhang, G. (2005). An extended Kuhn–Tucker approach for linear bilevel programming. *Applied Mathematics and Computation, 162*(1), 51–63.

Steinhofel, K., Albrecht, A., & Wong, C. K. (1999). Two simulated annealing-based heuristics for the job shop scheduling problem. *European Journal of Operational Research, 118*(3), 524–548.

Taillard, E. (1994). Parallel taboo search technique for the job shop scheduling problem. *Journal on Computing Science, 6*, 108–117.

Tupia, M. (2004). Un algoritmo GRASP para resolver el problema de la programación de tareas dependientes en máquinas diferentes. In *Proceedings of Conferencia Latino Americana de Informática CLEI (30, 2004, Perú)* (pp. 129–139).

Tupia, M., & Mauricio, D. (2004). Un algoritmo voraz para resolver el problema de la programación de tareas dependientes en máquinas diferentes. *RISI, 1*(1), 9–18.

Valero, S., Argente, E., Giret, A., Julian, V., & Botti, V. (2005). *Goodness and Lacks of MAS Methodologies for Manufacturing Domains, 3690*, 645–648.

van Laarhoven, P., Aarts, E., & Lenstra, J. K. (2002). *Job Shop Scheduling by Simulated Annealing Operations Research, 40*(1), 113–125.

Verter, V., & Dincer, M. C. (1992). An integrated evaluation of facility location, capacity acquisition, and technology selection for designing global manufacturing strategies. *European Journal of Operational Research, 60*(1), 1–18.

Walton, C. (2006). *Agency and the semantic web*. Oxford University Press Inc.

Wan, S., Yang, F., & Izquierdo, E. (2009). Lagrange multiplier selection in wavelet-based scalable video coding for quality scalability. *Signal Processing: Image Communication, 24*(9), 730–739.

Wooldridge, M. (2000). *An introduction to multiagent systems*. Chichester, England: John Wiley & Sons.

Wooldridge, M., & Jennings, N. R. (1995). Intelligent agents: Theory and practice. *The Knowledge Engineering Review, 10*(2), 115–152.

Wu, S. H., Fuh, J. Y. H., & Nee, A. Y. C. (2002). Concurrent process planning and scheduling in distributed virtual manufacturing. *IIE Transactions, 34*, 77–89.

Yu, J., & Buyya, R. (2006). Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. *Scientific Programming, 14*(3), 217–230.

Zambonelli, F., Jennings, N. R., & Wooldridge, M. (2003). Developing multiagent systems: The Gaia methodology. *ACM Transactions on Software Engineering and Methodology, 12*(3), 317–370.

Please cite this article in press as: Zato, C., et al. Model for assigning roles automatically in egovernment virtual organizations. *Expert Systems with Applications* (2012), doi:10.1016/j.eswa.2012.01.185