Resource Management in an Agent-based Virtual Organization — Introducing a Task Into the System

Michal Szymczak, Grzegorz Frackowiak, Maria Ganzha, Maciej Gawinecki, Marcin Paprzycki
System Research Institute, Polish Academy of Sciences, Poland

Myon-Woong Park
Korea Institute of Science and Technology, Seoul, Korea

Abstract

In this note we present a top level overview of resource management in an an agent-based virtual organization. To illustrate its main features we discuss processes involved in a project being introduced into the organization.

1. Introduction

Let us consider an organization in which teams of researchers are engaged in R&D projects and share a common virtual work-space. Obviously, team work requires cooperation between members and support of collaborative research has to go beyond, even most sophisticated forms of, document versioning and flow of resources in the hierarchical structure of the organization. What needs to be taken into account is: (1) representation of domain specific knowledge—to provide context for management of resources pertinent to the project (e.g. establishing a specific "location" of a resource within the domain knowledge allows for resource indexing, clustering and proliferation); (2) representation of structure of interactions and flow of resources in the project—to route resources based on project needs and responsibilities of team members; (3) representation of user profiles (situated within the domain knowledge and the structure of the project)—to specify team member *interests*, *needs* and *skills* (e.g. to establish what to do with new/incoming resources); (4) adaptability of the system—to deal with the fact that as the time passes the domain of scope of the project may expand, contract or shift; functional interrelationships between team members can change; their interests, needs and skills may evolve; and, team members may be added, removed or replaced.

It is relatively easy to see that these four points can be generalized beyond the initial collaborative research scenario. Let us assume that for the second point we utilize a notion of a virtual organization [8, 9, 10, 11, 12, 13], which allows us to define roles, interdependencies and interactions of participants. In such an organization its members need access to resources (here any entity within the organization, human and non-human, is considered a resource) to complete their individual tasks and to facilitate completion of group assigned projects. Access to such resources should be, among others, adaptive (change with the task) and personalized (each team member requires access to different resources; where access may be restricted by the organization policy/structure). Development of a system that would facilitate such functionalities is the aim of our project. In this context, in this note, we focus on processes involved when a new task/project is introduced into an organization; approached from the point of view of resource management.

2. Proposed system

Resource management is at the core of our approach and it is very often claimed that the best technique for resource representation is ontological demarcation (see for instance [7]). In this context, representation and management of resources (including their flow) can be achieved as a result of a two-step process. First, roles of participants are specified, and second, the real-world organization is represented as a virtual agent-based system (where each person is represented by its *Personal Agent* (*PA*) that can play different roles in different situations (e.g. in one project be a manager, while in another be a quality assessor), and a number of auxiliary agents is added to facilitate functioning of the system). For human resources, agent roles and interactions are combined with domain

ontologies (specifying the "area of operation," as well as context for structuring resources in the organization), while for other resources only domain ontologies are used. In both cases an overlay model allows specification of profiles of individual resources (see [1, 5, 6, 3]).

Let us add that ontological representation of resource profiles supports various forms of automatic reasoning (resource matching, query rewriting etc.). Furthermore, ontologies, overlay-based profiles and agent systems are naturally adaptable. Ontologies can be easily modified (e.g. extended) while adaptation of overlay-based resource profiles involves changes in weight associated with individual features. Finally, changes in virtual organization are easily achieved through changes in patterns of agent interactions ([4]).

We can now summarize the fundamental features of our approach to building an environment for supporting context aware personalized resource provisioning within a virtual organization:

1. *Domain knowledge* is represented in terms of ontologies.

2. *Organizational structure* is decomposed into interacting agents.

3. *Overlay model* is used to represent resource profiles.

4. *Resource matching* utilizes semantic reasoning involving resource profiles.

5. *System adaptability* is reached through adapting: (a) pertinent ontologies, (b) resource profiles, and (c) structure of the agent system.

6. *Human resource adaptability* is achieved by (e-)learning ([2]).

3. Introducing and managing a task in the system

Let us now present birds-eye view of the system, by discussing processes involved in introducing and running a project. Specifically, we consider two (very different, though both representable in our approach) scenarios: (1) customer requesting from an IT organization creation of an intranet and a company portal; (2) cable TV installation ordered from a cable network company. To focus our discussion, in Figure 1 we present the use case diagram of the system. Unfortunately, due to the lack of space we focus only on selected key functionalities. Note also that the following discussion is written in terms of *units with specific roles*, and such units can consist of one (or more) humans, agents, or "teams" consisting of humans and agents.

When a service/project is requested from an organization (which can be anything from a one-person company to 50,000+ employees corporation) a *Project Manager* (*PM*) is associated with it. Its first task is to make sure that the request is thoroughly analyzed and on the basis of such an analysis to make a decision if a job should be accepted. This task is delegated to the *Analysis Manager* (*AM*). At the same time a *Task Monitor Agent* (*TMA*) is created to oversee the task performed by the *AM* (for more details about role of the *TMA*, see below). It should be noted that the structure of the *AM* can be either very complicated and consist of a number of humans and agents (in the case of a corporation) or very simple (in case of a small business). Finally, it is even possible that the *PM* can play the role of the *AM* (e.g. in the case of a very small business or self-employment). Most important deliverable prepared by the *AM* is decision support for acceptance or rejection of the request. The report prepared by the *AM* is backed up, among others, by the cost, resource and income analysis.

Since we assume that all profile data processed in the system is based on appropriate ontologies, one of crucial tasks of the *AM* is to "translate" the common language requirements originating from the user into requirements constructed utilizing ontologies employed in a given organization. Fulfilling this need, at the beginning of its work, the *AM* instantiates a new resource called *Project Request* (which has its own profile). This resource is used when the *AM* performs initial automatic analysis and creates first version of the *Project Requirements Document* which, again, is a resource. During its work the *AM* analyzes resources available in the organization (their profiles, availability and accessibility). For instance, in the case of cable TV installation this step should be rather simple and involve steps like checking whether the customer who requests installation lives in the area covered by the service provider and if there are resources available to install the cable TV within a specific time-frame. Such an analysis could easily be performed by a software agent with a build-in expert system. On the other hand in the scenario of the intranet and portal, analysis would involve more elaborate actions, such as: checking technological requirements for the project, existing similar solutions, organization or customers experience etc. In this case the *AM* would most likely involve human resources. As a result of its work, the *AM* prepares the *Requirements Analysis Document*. Let us mention, that in case of a corporation the *AM* may not have permission to see all the resources available in the organization, but in this

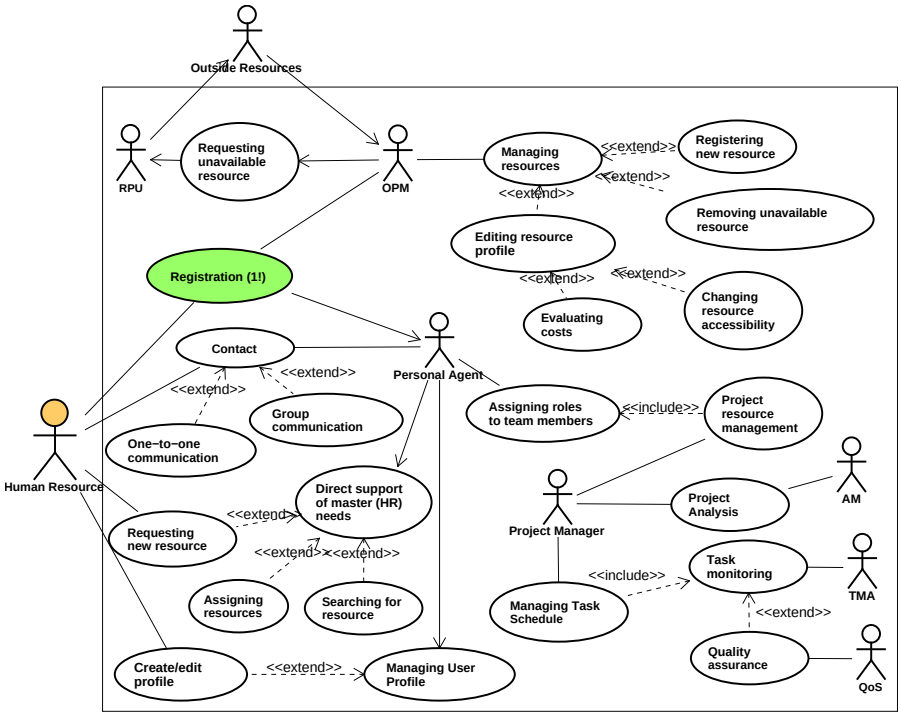

Figure 1. Use case of the system

case it will specify resources that the project needs and which, according to its best judgment, are unavailable.

If the *AM* recommends that the project is rejected, and the *PM* concurs, user is informed about the decision and this ends the process. Let us now assume that the project was accepted. As a result the *PM* prepares the *Project Schedule* and on its basis works to assure *Resource Reservation* (note that the fact that John, the Java coder, works for organization *X* does not mean that John is available starting from next Monday). Before the schedule is constructed the *PM* has to analyze available resources (its own and provided by the customer). It may involve checking availability of programmers who have the required competence in PostgreSQL, object oriented programming and recent web technologies, as well as availability of resources such as: servers, (e-)learning materials for software to be used in the project, licenses and requirements for both test and final deployment environments etc. Again, the *PM* analyzes only these resources which it has access to. If resources that the *PM* knows about are not sufficient, the *PM* requests the *Organization Provisioning Manager* (*OPM*) to facilitate the missing resources (e.g. Java programmer(s), or a DB2 e-learning course). Note that such resources may be available in the organization, but the *PM* may not have access to this knowledge. *OPM*'s role is to provide resources for other resources which request them. To fulfill its role, we assume that it has access to information about all resources available in the organization. Since the *OPM* can be queried by (authorized) resources that play various roles in the organization, it has to analyze available resources using various patterns of reasoning and possibly some expert systems. Note also that, again, the *OPM* can be either an agent, a human represented by its *PA*, or a composite structure consisting of multiple agents and humans (e.g. it can have a resource that indexes and routes incoming documents / books / journals, a search engine, a library material acquisitor, etc.). Again, if the resource (a) is found, and (b) can be reserved (for a specified time), it is then assigned to the requesting *PM*. Otherwise, the *OPM* triggers action of a *Resource Procurement Unit* (*RPU*), which is responsible for finding an appropriate resource. Assuming, for instance, that Java programmers and DB2 e-learning materials were not found in the organization the *OPM* will generate a request to the *RPU* to acquire specific (ontologically demarcated) resources. The *RPU* in turn will communicate it to the "world outside of our system." Obviously, specific deliverables produced by the *AM* and the *PM* differ depending on the organization and the project, but the structure of the process remains the same.

Let us now address processes that take place after the *Project Schedule* is created. Note that this may involve communication between the project *PM* and other managers within the hierarchical structure of the organization as well as the user; resulting in acceptance of the final version of the *Project Schedule*. Let us also stress that the *Project Schedule* is a resource itself and has its own profile. It is used by the *PM* to define tasks and assign them to appropriate (human or non-human) *Resources* (*R*). Note that each *R* can be either a single resource, or a collection of resources treated a single unit. For instance, team that is responsible for the back end of the portal may consist of 4 coders and a manager, while the team dealing with user interface could consist of 2 coders and an artist, etc. Both teams will be treated as a single resource, with its own task, and a *Task Monitor Agent* (*TMA*) associated with it. At the same time, inside these composite resources the proposed organizational structure will be repeated, and individual tasks and their *TMA*s instantiated. It is by communicating with *TMA*s the *PM* can monitor status of all tasks (including their start and completion), add task, risks and alerts.

The *TMA* monitors a given task until its completion (then it is killed by the *PM*). While working on the task, process *R* might be interrupted by unexpected circumstances which either it can deal with (e.g. finding tips on how to deal with heap memory exceeded error in Java, or how to build a DB2 cluster) or ones that will probably influence other parts of the project (e.g. customer requested that a different data structure is to be interfaced with, or some additional unavailable resources turn out to be needed, or a particular employee has to immediately take a leave, etc.). These circumstances are expected to involve *PM*'s reaction and should be tagged appropriately in the *Task Monitor Agent*. Note also that not every interrupt requires *PM*'s intervention. We assume that resources can interact with each other (which resource can communicate with which one is specified by the organization), among others, to solve basic problems occurring during task execution. For instance to find a manual for software used in the project given *R* can contact other *R*s in its group. Finally, each *R* might generate multiple interrupts, but as long as these do not require the *PM* to react they are going to be tackled locally.

Obviously, at a certain moment given task comes to an end. Upon completion of a task, the task-specific *Quality of Service* (*QoS*) module analyzes the work. A *QoS* module might be a team of humans, as well as an expert system, a simple unit test for Java code, or TV signal test after the TV is installed. Unless the quality of the work is not satisfactory and further improvements are needed, the *PM* is informed about the tasks completion. If the result of the task does not fit the requirements there is a necessity to repeat some part of, or even the whole task. This can take more time and resources than it was specified in the *Project Schedule*. However, only conflicts with the schedule should result in the *PM* being alarmed. Note that a 'major interrupt" that results in changes in the *Project Schedule* may need to be propagated within the structure of the team that works on the project.

Obviously, completion of a (sub)task may trigger execution of another (sub)task specified in the workflow of a given project. Upon completion of all (sub)tasks specified in the *Project Schedule*, the project is completed.

4. Agents in the system

Previous section described an abstract business process within an virtual organization. According to it, we distinguished roles that are played by *PM*, *AM*, *R*, *RPU*, *OPM TMA*, and *QoS*. As noted above, in some cases these roles will be fulfilled by software agent(s), some of them are likely to be played by one or more humans (supported by their *Personal Agents*), while some are likely to be completed by a team consisting of software agents and humans. While specific arrangements may depend on the particular organization (and its domain of operation), the process described above remains unchanged. Note also that we have identified a few situations that are expected to trigger reaction of a human actor:

1. requirements analysis assessment
2. accepting a particular person to become a manager of a project
3. changes in customer requirements
4. the *OPM* cannot find required resource within the organization
5. accepting *Resource Reservation* document
6. negotiating and accepting *Project Schedule*
7. final task acceptance

Even though human intervention is likely to be required, our interest is in performing as many tasks as possible either in an autonomous fashion or to provide support for humans in fulfilling the above specified roles. To this effect we utilize software agents. The role-based approach allows us to specify sets of functions associated with each role. These functions may

be then undertaken by autonomous agents that fulfill a given role (e.g. a *TMA* agent that makes sure that Cable TV was installed before 11:45 at a specific address and if this is not the case, raises an alarm), or by a *Personal Agent* (*PA*) that helps the human *PM* in managing a team of coders. The process is as follows: the autonomous agent, when created to fulfill a given role is provided with required modules to accomplish it, e.g. the *TMA* obtains information about the deadline it is to observe and what to do in it is, or is not met. The situation is somewhat more complicated in the case of the *PA*. First, let us note that we envision that every worker is represented in the system by her/his own *PA*. Furthermore, upon joining the system (and thus the organization) the *PA* registers with the *OPM*—the resource manager—and becomes one of available resources. In Figure 1 we have depicted the *PA* and conceptualized it as an interface between the human and the remaining parts of the system; as well as a "helper" that supports user in fulfilling her role. Since role can change, the *PA* has to be able to support user in anyone of them. However, in Figure 1 we were able to identify core functions of the *PA*, which are used regardless of a specific role. To provide support for the user who is assigned a specific role, modules facilitating functions associated with that particular role are then loaded into the *PA*, extending its functionality. Note that in the, somewhat more complicated case, when a team of analysts (*AM*) estimates feasibility of a project, each team member will be represented by its *PA*. Therefore, a role-specific set of interactions between these *PA*'s and humans they represent will constitute fulfillment of the role *AM*.

Observe that in this way, we have only "one basic type" of an agent in the system: the *Personal Agent* (supplemented by possibly some auxiliary agents). The *PA* can support user in playing each role identified in the organization. This in turn matches very nicely with the real world organization, where we also have only one type of entity: human being that can play various roles identified in the organization.

5. Concluding remarks

In this note we have presented birds-eye view on the agent-based virtual organization in which the domain of interest and the structure of the organization itself are ontologically demarcated. Furthermore, the information management is also ontology-based. The main goal of this note was to functionalize processes involved in a task being introduced into the organization. We believe that above described interactions and activities can serve as a basis for more precise definition of a virtual organization facilitating system. The first step in this direction will be creation of initial ontologies of: (1) specific selected domain, (2) profile of resource in that domain, (3) organization, and (4) profile of resource in such an organization; which is what we are working on at present.

Acknowledgment

This work is partially sponsored by the KIST-SRI PAS "Agent Technology for Adaptive Information Provisioning" grant.

References

[1] Fink, J., Kobsa, A.: User Modeling for Personalized City Tours, *Artif. Intell. Rev.*, 18(1) (2002) 33–74.

[2] Ganzha, M., Paprzycki, M., Popescu, E., Badica, C., Gawinecki, M.: Agent-Based Adaptive Learning Provisioning in a Virtual Organization. Proceedings of the AWIC Conference. In: Wegrzyn-Wolska K.M., Szczepaniak P., (eds.), Advances in Intelligent Web Mastering, Springer, 2007, 112-117

[3] Gawinecki, M., Gordon, M., Paprzycki, M., Vetulani, Z.: Representing Users in a Travel Support System. In: Kwaśnicka, H. et. al. (eds): *Proceedings of the ISDA 2005 Conference*, IEEE Press (2005) 393–398

[4] Gawinecki, M., Gordon, M., Nguyen, N.T., Paprzycki, M., Zygmunt Vetulani, Z.: Ontologically Demarcated Resources in an Agent Based Travel Support System. In: R. K. Katarzyniak (ed.): *Ontologies and Soft Methods in Knowledge Management*, Advanced Knowledge International, Adelaide (2005) 219–240

[5] Kobsa, A., Koenemann, J., Pohl, W.: Personalised Hypermedia Presentation Techniques for Improving Online Customer Relationships, *Knowl. Eng. Rev.* 16 (2001) 111–155

[6] Montaner, M., López, B., de la Rosa, J. L.: A Taxonomy of Recommender Agents on the Internet. In: *Artif. Intell. Rev.* 19 (2003) 285–330

[7] Fensel, D., Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce, Springer, 2001

[8] Warner, M., Witzel, M.: Zarzadzanie organizacja wirtualna, Oficyna Ekonomiczna 2005

[9] Barnatt, C.: Office Space, Cyberspace and Virtual Organization, "'J. of General Management"', (1995) 20(4), 78–92

[10] Bleeker, S.E.: The Virtual Organization. In: Leading Organizations, ed.: Hickman, G.R., Sage, 1998

[11] Grenier, R., Metes, G.: Going Virtual: Moving Your Organization into the 21st Century, Prentice-Hall, 1995

[12] Goldman, S.L., Nagel, R.N., Preiss, K.: Agile Competitors and Virtual Organizations, Van Nostrand Reinhold, 1995

[13] Dunbar, R.: Virtual Organizing. In: International Encyclopedia of Business and Management, ed.: Warner, M., Thomson Learning, 2001, 6709–6717