

Dynamic Generation of User Communities with Participative Knowledge Production and Content-driven Delivery

Sinuhé Arroyo
Intelligent Software Components, S.A.
sinuheag@isoco.com

Juan Manuel Dodero
Universidad Carlos III de Madrid
dodero@inf.uc3m.es

Abstract

In a distributed knowledge management system, knowledge is firstly produced and then delivered to a person or community of users that is interested in it. Knowledge creation or production is a set of cooperative tasks that need to be coordinated. A multiagent architecture is introduced for this aim, where knowledge-producing agents are arranged into knowledge domains or marts, and a distributed interaction protocol is used to consolidate knowledge that is generated. The knowledge that is produced in this way is used as the source data to dynamically build user communities that can drive the delivery of knowledge amongst users.

1 Introduction

Knowledge Management (KM) is the group of processes that transform intellectual capital of an organization or group of persons into a value [18]. Amongst those processes, KM authors quote creation, acquisition, distribution, application, sharing and reposition of knowledge [17]. Knowledge emerges from the social interaction between actors, and frequently it is not formally structured to be appropriately used and exploited. In this sense, knowledge management acts as a systematic programme to profit from what the organization knows. KM processes can be summarized in production, acquisition and delivery of knowledge. On the one hand, acquisition and delivery tasks allow to share and reuse the group-wide available knowledge. On the other hand, production is a creative process to formulate new knowledge in the group, that has to be validated according to the definition of knowledge described below.

When a group of people is participatively creating (or producing) a complex object, it is advisable to establish a set of rules to coordinate its development. This is the situation, for instance, when several people are building a software object. We will follow a concrete example for a better explanation of the problem. Let's suppose two developers who are designing respective modules, which will be part of the same software object. During the design, the necessity to develop two sub-elements for the same purpose can be detected by both developers. Each one usually has his/her own pace of work in developing the common element. Also, they can be differently skilled in that work. If the development process is not appropriately coordinated, the following problems can arise:

- A developer could get her work crushed, depending on the required speed and quality of the design, in comparison to her partner's competency.
- When speed is more important than quality, a more elaborated and reusable product can be readily thrown away.
- In the best case, effort will be duplicated in several phases of the project.

Therefore, the coordination of participative production of knowledge meets the following objectives:

- Bring together participants' different pace of creation.
- Take advantage of participants' different skills in the problem domain and the tools that are managed.
- Reduce the number of conflicts provoked by interdependencies between in-production knowledge components.
- In a more general sense, avoid duplication of effort.

On the other hand, knowledge needs to be constantly updated and delivered to the right places at the right time. Knowledge delivery is a continuous process of transferring knowledge that is interesting to a person or community of users. These and other questions are raised in order to distribute knowledge among interested users:

- Which communities are users member of?
- How are the communities initially spawned, after setting up a group of users who are generating knowledge?
- What would it happen if a user changes the kind of knowledge that it produces, and this is better classified in another community?
- As time progresses, may knowledge that is produced in a community be biased towards a different category?

2 Agent-mediated knowledge production

Several authors on Knowledge Management cite production or generation of knowledge referring to the creation of new knowledge [3][15][17]. When Davenport and Prusak tell about *knowledge generation*, they are referring both to externally acquired knowledge and that one developed within the bosom of an organization, without making any difference between acquisition and generation. In our study, we consider *generation* as distinct from *acquisition*. From our point of view, knowledge generation or production is the creation of new knowledge as the result of the social interaction between actors in a workgroup or organization, according to their interests and the regulations that apply. On the other side, knowledge is acquired when it comes from outside of an organization or workgroup —i.e., it is generated outside and thereafter adopted by the organization.

Coordination is a key pattern of interaction that is needed to obtain a good-quality knowledge that has been validated by means of contrast and/or consensus in the group. Although distributed KM research in knowledge acquisition and sharing efforts are worth to be considered, knowledge production still lacks interaction models and methods to coordinate a group of autonomous users in the cooperative generation of knowledge.

Multiagent systems have been successful in the distributed implementation of KM processes. Knowledge acquisition agents have been one of the most successful applications of software agents, specifically in the Internet [7], where knowledge-collector agents operate within available information resources, and validate them in accordance with the users' interests. On the other hand, knowledge delivery lies in an end-to-end routing of knowledge that is generated by some actor, and it is another typical task that has been realized by software agents [9][10]. Therefore, it is reasonable to approach the multiagent paradigm for knowledge production. Knowledge-producing agents need to do formulations that keep with a validation scheme supporting the knowledge construction. Since agents have been proven as a helpful tool for the coordination of human people who are performing a given task [13], multiagent systems can support the coordinated interaction needed to achieve an agreement on the knowledge that is eventually generated, and even the validation scheme.

Agent interaction protocols govern the exchange of a series of messages among agents, i.e. a conversation. There are some popular interaction protocols and architectures, used heavily by multi-agent systems, like blackboards [14], contract protocols [16] and computational economies [19]. Nevertheless, these approaches tackle rather general aspects of agent interactions, usually characterized as competitive, cooperative or negotiative. During knowledge production, agents try to convince each other in a group to accept a given knowledge in some domain, so building the corpus of shared knowledge. The aim is to allow agents to *consolidate* knowledge that is continuously produced. Consolidation in a group of producers is to establish a given knowledge as accepted by the group as a whole, with every member knowing about that circumstance. Agents can reach a consensus on the knowledge that is consolidated by the exchange of messages, using the consolidation protocol described below.

The architecture and protocol presented below is a multi-agent approach to the production of knowledge. The working hypothesis is that a group of agents can help in the participative production of knowledge, by coordinating their creation activities. Therefore, different agents can act as representatives of knowledge-producing actors, according to the following principles:

- Agents can be structured into separable knowledge domains of interaction. This structuring reflects the knowledge differences between producers.
- A dynamic re-thinking of the structure of interactions in different domains can help to reduce the

inter-dependencies during the process.

2.1 A multi-agent Architecture for Knowledge Production

In our architecture, knowledge-producing agents can operate within the boundaries of a specific domain or knowledge mart, as shown in figure 1. Nevertheless, interaction among different domains is also supported through a number of proxy agents. In order to facilitate interaction between domains, marts can be structured in a hierarchical way. In this architecture, domains can be modelled as knowledge marts, and marts are arranged into knowledge warehouses. A Participative Knowledge Mart (PKM) is a distributed group of agents that is trying to produce a piece of knowledge in a given domain. A Participative Knowledge Warehouse (PKW) is the place where knowledge produced in foreign marts is merged in a structured fashion.

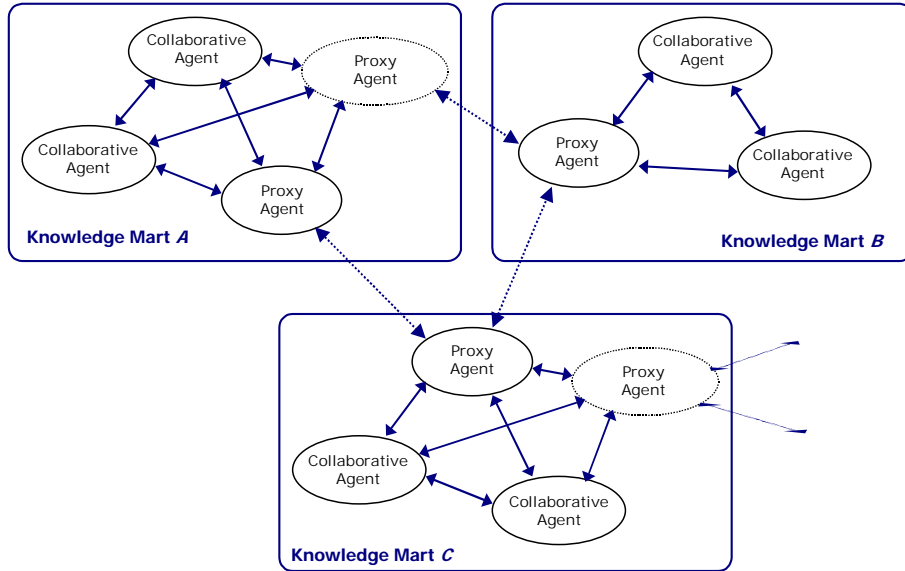


Fig.1. Participative knowledge marts

Two or more PKMs can interact using representatives in a common PKW. When knowledge produced in a mart can affect performance in some other domain, a special proxy agent can act as representative in the foreign mart, according to the proxy design pattern [8], so that interaction between marts is not tightly coupled.

2.2 Knowledge Consolidation Protocol

The function of the protocol executed by agents is to consolidate knowledge that is created in our agent-coordinated interaction environment. By *consolidation* we mean the establishment of knowledge as accepted by every agent in the mart, in such a way that every member agent eventually know about it. The consolidation protocol, described in [5], is a two-phase process:

- The *distribution* phase begins when an agent submits a proposal, i.e. when the agent starts the protocol. A given timeout t_0 is defined to set the end of this phase.
- The *consolidation* phase begins if there is a proposal waiting to be consolidated. This event can occur whether the distribution timeout t_0 expired or a t_0 -waiting agent received a proposal that was evaluated as preferred. A distinct timeout t_1 is used for the consolidation phase.

An agent can participate in several interaction processes. Each interaction process is handled separately, by initiating a new execution thread of the protocol. Two different timeouts are used over the course of the protocol. Timeout t_0 is used for the distribution phase, that occurs after an agent submits a *proposal(k,n)* message, where n is an interaction process and k is a piece of knowledge that wants to be consolidated in the PKM. During t_0 , messages can arrive from any other agent, consisting in new proposals, referring to same interaction process. Timeout t_1 is used for the consolidation phase, that

occurs if there is a proposal waiting to be consolidated (this can occur whether t_0 expired or a t_0 -waiting agent received a proposal that was evaluated as preferred). The message used to consolidate a proposal has the form *consolidate(k,n)*, and its aim is to establish a previously submitted proposal k as accepted in an interaction process n .

At any moment, the reception of a message from another agent may provoke a momentary retraction from a previously submitted proposal, until a counter-proposal is elaborated. An agent that has not reached this state will be waiting for t_0 timeout. Then, if the agent receives a proposal that is evaluated as preferred, a new timeout t_1 is set to give it a chance. But if the preferred proposal is not eventually ratified, then the agent goes on about its aims and will try again to consolidate its own proposal.

Agents' rationality needs to be modelled in terms of preference relations or relevance functions, in order to allow them to evaluate and compare proposals. The relevance of a proposal is defined as the set of proposal attributes considered when interacting, while the preference relationship denotes which of two proposals is preferred.

The sequence of events spawned by the execution of the protocol by two agents trying to consolidate their proposals at the same time is depicted in the figure 2. The interaction begins when agents A_1 and A_2 submit proposals p and q respectively.

- (a) Both A_1 and A_2 receive each other's proposal and begin the distribution phase, so starting timeout t_0 . Proposals p and q also arrive to A_3 , which is not participating in the process and silently receives them.
- (b) A_1 compares q to p , turning out that its proposal has a worse evaluation. It is reasonable that an evaluation of proposal p obtains a higher value than q , as for the second objective described above. Concerning first and third objectives, any relevance function should outcome similar values for both proposals, so they would not be decisive. Then, A_1 starts timeout t_1 , giving q a chance to be consolidated. On the other hand, A_2 also compares both proposals and reminds A_1 of the results by sending again q , then extending timeout t_0 in order to give a chance for other agents' proposals to come.
- (c) When timeout t_0 expires, A_2 sends a consolidation message for q that arrives to every agent in the mart. At the reception, A_1 finishes the protocol because it is expecting the consolidation for q . A_3 simply accepts the notification.
- (d) Finally, at the expiration of t_1 , A_2 is confirmed about the end of the consolidation phase for q and its execution of the protocol finishes successfully. Therefore, every agent in the mart will eventually know about the consolidation of the proposal.

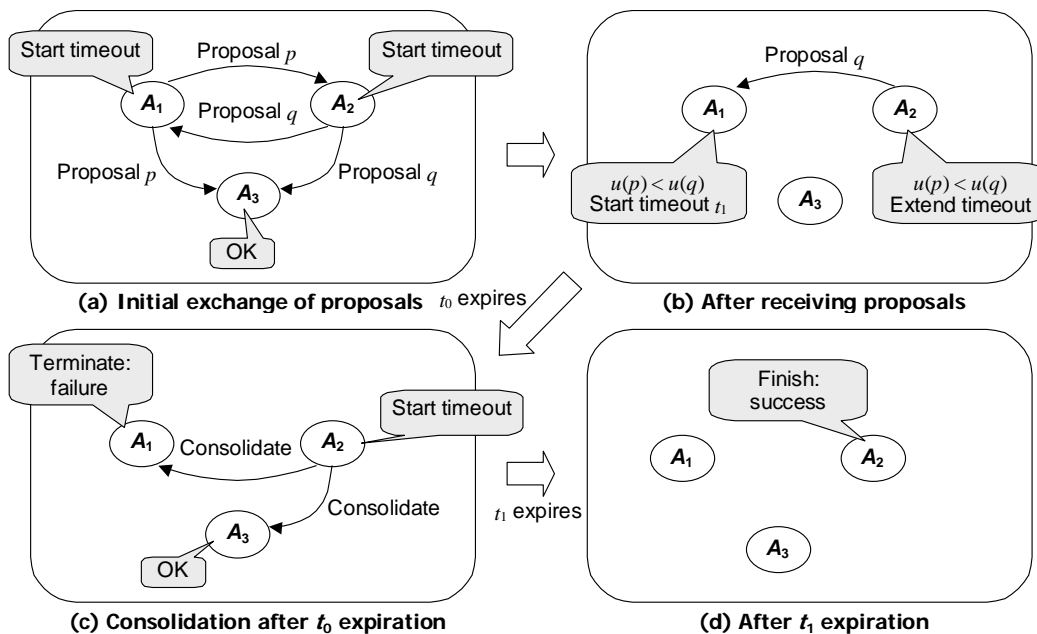


Fig. 2. Execution example of the protocol

3 Content-driven Knowledge Delivery

Bringing forward the answers to knowledge delivery issues, it seems reasonable to dynamically establish the membership of agents into marts. As well, division and/or fusion of marts can be needed to better reflect the knowledge-directed proposed structure. To represent the differences between knowledge marts, we define a *cognitive distance* between two agents as a measure of the similarity between the knowledge produced by both agents. The cognitive distance can be also defined between two marts, in the sense that these are dynamically formed groups of knowledge-carrying agents. In that case, clustering techniques can be readily applied to solve the above-mentioned problem of dynamic membership of agents into marts. Data about the cognitive distance between marts can be taken from agents' activity logs. For instance, a web server log file is a rich data source to determine a cognitive distance.

3.1 Content-Driven Information Delivery

A successful building of a complex object requires the coordination of ideas that are exchanged. Due to the heterogeneity of groups of people and environments in which this building takes place, reducing the coordination and communication efforts should be a major task [12].

A way to achieve this goal is to use an *event service* to deliver information based upon its content. The service will observe the occurrence of events or combination of events and will notify the members of the group, who have previously shown their interest in reacting upon a concrete occurrence. Such an interest is established by the collaborative agents (members of a knowledge production group) through filters upon the contents produced in the knowledge mart. The main advantage of this approach resides in the expressiveness that may be obtained in the communications, since the filters apply to the whole content of the notification, allowing a greater degree of freedom in the way the information is codified [1]. Filters are used to deliver contents of interest to the appropriate recipient.

Addressing is concern of the underlying transport protocol that guarantees a reliable delivery, and can be done using some multicasting facility in the underlying transport. The main disadvantage of multicasting is the loss of expressiveness, due to the necessity of mapping expressions to IP groups in an scalable way [1]. Since multicasting never relates two groups of different IP addresses, a notification that matches two or more filters, corresponding to agents located in different groups of IP addresses, should be routed in parallel with the rest, so reducing the protocol effectiveness and increasing the communication efforts.

A workaround to overcome that issue is to use a set of connected brokers, which can also work as entry points for collaborative agents. Such brokers will be responsible for the routing of notifications or, in the last case, to deliver them to other agents. In order to do addressing, each brokers holds a table with filters, to send notifications matching a concrete filter to the appropriate addresses. Eventually, when notifications arrive to a broker who acts as an entry point for some agents, they will be forwarded to those whose filters are matched [2].

3.2 OKAPi as a Content-Driven Knowledge Broker

OKAPi (Ontology-based Knowledge Application Interface) is an implementation of a multiprotocol adapter for a content based-router. The main purpose of the system is to facilitate the request delivery of information in WAN/LAN environment independently of the communication protocol (HTTP, SMS, SMTP, etc). There is no limitation to the interoperability of different protocols, and so, an agent may establish some filter using HTTP, and receive notifications by SMTP or SMS, whilst the publisher used a third party protocol to explicit the contents.

The filters to select the contents of interest are established using a an ontology as a vocabulary and reference model, in order to provide every agent with a shared way of communication within the domain of a concrete complex object.

Agents play one of two very well-defined roles in OKAPi, according to the necessity or availability of information showed by the agent:

- **Subscribers:** Agents who want to gather some information referred to a particular complex object. They are responsible for the establishment of filters using the appropriate ontology corresponding to a particular object.
- **Publishers:** Agents that make some information explicit to the rest of the community by sending

it to the broker that acts as its access point to the service.

Both roles are not mutually exclusive, and in some cases an agent can act as a subscriber for some complex object, and as a publisher for another.

A filter is said to be covered, when for every one of its attributes there is a publication which holds values that accomplishes all the properties of the filter, according to the logical operator ($=$, \leq , \geq , etc.) defined for each. Once a filter has been set, it is propagated to every broker in the knowledge mart that holds an interested agent, in a least-information transferring basis. This means that if a broker situated upstream holds a more general filter that contains the current one, it won't be forwarded any further that way, since all the publication that fit the former will fit the current subscription. This mechanism guaranties that an agent is part of all the knowledge marts it is interested in, and so, that the contents produced within the interested domains will be forwarded from one router to another, following the reverse return path established by the filter, until it reaches the appropriate access point. This is true for every agent interested in that particular knowledge. From the access point, the contents of the notification are formatted according to the agent's desires and sent using the addressee's protocol (SMTP, SMS, HTTP, etc.)

4 Dynamic Generation of Agent Communities

In order to obtain a more effective communication mechanism, the interest of agents that make up the service (subscriptions and publications) are analysed using clustering techniques. As a result, agents that share similar interest (sharing the same PKM) are relocated into communities located in the same access point, or close to those that at the moment are of their interest, reducing in this way the amount of information that brokers must share to complete de service.

As mentioned earlier, an agent may take two different roles, either a subscriber or a publisher. In both cases it holds some degree of interaction with a concrete mart, whether it is producing or is interested in receiving some content. For this duality, the affiliation of an agent to a mart has to take into account both circumstances.

OKAPi provides a mechanism to overcome the agent relocation based on the amount of knowledge produced within a mart. The biggest the amount of knowledge produced among agents (number of subscriptions, publications sent or received), the biggest the probability that they are relocated closer to each other with respect to the marts they are interacting with. Such probability is measured as the number of messages they interchange as reflected in the servers' logs (cognitive distance). According to this, if the relocation of agents to a new access point takes place eventually, the chance that they send or receive messages to/from the mart or marts they were member of will be smaller than the chance of receiving or sending within the current one.

A more specific case of agent relocation could occur when not only some agents of the community are relocated, but the hold community is resettled. Such relocation could be originated by three different circumstances:

- Significant reduction of the cognitive distance among marts. The contents produce by some marts are very much alike, which suggests that they could be merged into a single one.
- A change in produced contents: The knowledge produced can change, and it seems clear that the previous thread is left aside due to obsolescence, or just lost interest.
- There are more than one very well defined thread of contents: It could happened that the knowledge follows well defined but divergent paths within the mart. If that is the case, the mart should be divided and the resulting knowledge groups relocated in different access points.

All these issues can be addressed by means of clustering techniques, but in some cases there are collateral considerations thought as fundamental for the appropriate functioning of the knowledge mart, as could be the optimal size for marts or the cost of agents relocation related to the reduction in the brokers communication efforts.

$$\begin{bmatrix} 0 & & & & & \\ d(2,1) & 0 & & & & \\ d(3,1) & d(3,2) & 0 & & & \\ \dots & \dots & \dots & \dots & & \\ d(n,1) & d(n,2) & \dots & \dots & 0 & \end{bmatrix}$$

Fig. 3. Dissimilarity matrix.

The agent's cognitive distance to each existing mart is represented by the matrix of fig. 3. By analysing these distances, an optimal agent's placement could be deduced. An accurate algorithm to obtain the dissimilarity matrix, and so, the agents' and mart's best access points location is COBWEB. It applies a hierarchical incremental method which uses an heuristic evaluation measure called *category utility* to determine the correct agent placement [11].

Figure 4 represents an example of agent relocation due to the birth of a new thread of knowledge within the mart. In the first stage (left side) there are two marts composed by four agents each one of them. Due to a change of interest, a new mart is developed by the knowledge production of two agents. The cognitive distance among agents R and J have been reduced with respect to each other, but it did increase with respect to the mart, originating the creation of a new mart in a different access point. M and S are still part of the same community and the same happens to M and D. M is member of two marts, which means that it must be positioned in an access point not much far from those communities.

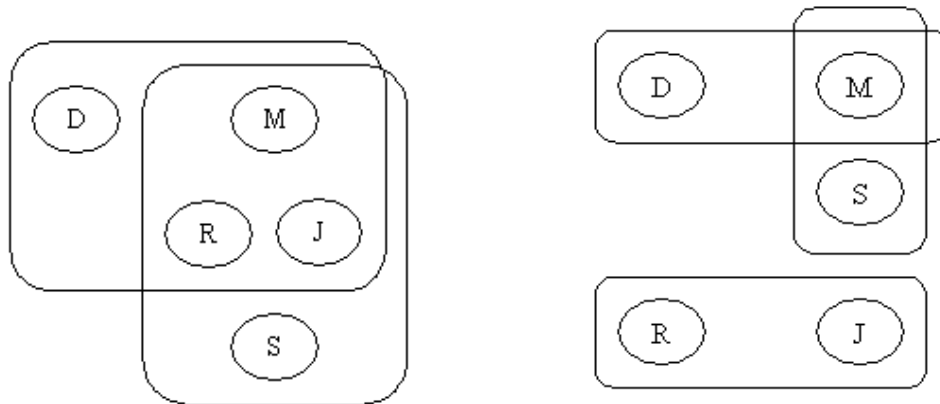


Fig. 4. Distribution of agents into communities

5 Conclusions

The participative approach presented in this work is applicable to several CSCW (Computer-supported cooperative work) tasks. It has been successfully applied to the instructional design of learning objects [4] [5] and electronic books composition [6]. Nevertheless, further validation is needed to assess the usefulness of the architecture in different scenarios. We are also conducting tests on the impact of the number of agents in the overall effectiveness of the model.

References

- [1] Carzaniga, A.: Architecture for an Event Notification Service Scalable to Wide-Area Networks. PhD Thesis. Politecnico di Milano (1998)

- [2] Carzaniga A., Wolf, A.: Content-Based Networking: A New Communication Infrastructure. NSF Workshop on an Infrastructure for Mobile and Wireless Systems. In conjunction with the International Conference on Computer Communications and Networks ICCCN. Scottsdale, AZ. (2001)
- [3] Davenport, T. H., Prusak, L.: Working Knowledge: How Organizations Manage What They Know. Harvard Business School Press. (1998)
- [4] Dodero, J. M., Sicilia, M. A., García, E.: A Knowledge Production Protocol for Cooperative Development of Learning Objects, Proceedings of the 2nd Workshop on Agent-Supported Cooperative Work, International Conference on Autonomous Agents, May 28-30, Montreal, Canada, (2001)
- [5] Dodero, J.M., Aedo, I, Díaz, P.: A Multi-agent Architecture and Protocol for Knowledge Production. A Case-study for Participative Development of Learning Objects. *Proceedings of the Informing Science 2002 Conference*, June 19-21, Cork, Ireland (2002) 357-370
- [6] Dodero, J. M., Aedo, I., Díaz, P.: Participative Knowledge Production of Learning Objects for e-Books, *The Electronic Library* **20** (2002)
- [7] Etzioni, O., Weld, D. S.: Intelligent Agents on the Internet: Fact, Fiction, and Forecast. *IEEE Expert*. (1995) 44-49
- [8] Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns. Reading, Massachusetts: Addison-Wesley Publishing Company (1994)
- [9] Genesereth, M., J. Tenenbaum, J.: An agent-based approach to software. Stanford University Logic Group. (1991)
- [10] Gruber, T. R.: Toward a Knowledge Medium for Collaborative Product. Proceedings of the 2nd International Conference on Artificial Intelligence in Design. Kluwer Academic Publishers. Pittsburgh (1992) 413-432
- [11] Han, J., Kamber, M.: Data Mining: Concepts and Techniques Morgan. Kaufmann Publishers (2001)
- [12] Liebowitz, J. (ed.): Knowledge Management Handbook. CRC Press LLC (1999)
- [13] Maes, P.: Agents that reduce work and information overload. *Communications of the ACM* **37**(7) (1994) 31-40
- [14] Newell, A.: Some problems of the basic organization in problem-solving programs. Proceeding of the Second Conference on Self-Organizing Systems. Spartan Books. (1962) 393-423
- [15] Nonaka, I., Takeuchi, H.: The Knowledge-Creating Company. Oxford University Press. New York. (1995)
- [16] Smith, R. G.: The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers* **29** (12) (1980) 1104--1113
- [17] Swanstrom, E.: Knowledge Management: Modeling and Managing the Knowledge Process. John Wiley & Sons (1999)
- [18] Stewart, T.A.: Intellectual Capital: The New Wealth of Organizations, Doubleday, New York, 1997.
- [19] Wellman, M. P.: A Computational Market Model for Distributed Configuration Design. Readings in Agents. Morgan Kaufmann. San Francisco, California. (1997) 371-379