SOFTWARE AGENTS

Gregor Stiglic, Mateja Verlic, Peter Kokol

University of Maribor, Laboratory for System Design

Agent technologies are considered as fundamental to the realization of next generation computing. A wide range of software engineering approaches have been devised to manage the complexity of distributed systems – among the most significant ones until now being object-orientation. Object-oriented programming provided abstraction level with the encapsulation of data and objects with several abilities including inheritance.

Now it is widely recognized that interaction plays crucial role in complex software. In most cases, software architectures contain many dynamically interacting components engaging in complex coordination protocols. Furthermore, with the changing requirements in systems' environments (mostly due to optimization of distributed systems for better performance) the need for a level of autonomy emerged. Autonomous components should be able to respond dynamically to rapidly changing circumstances in dynamic and open environments. Software agents take the next step after object-orientation by giving each 'object' the ability to be autonomous, reactive, pro-active and social – now the software components (agents) have their own thread of control, they respond to the changes in the environment, have their internal goals and are communicative. In distributed systems the idea of an agent is often seen as a natural metaphor.

Agent-based systems are systems in which the key abstraction used is that of an agent. An agent-based system can contain a single agent, but arguably the greatest potential lies in the application of multi-agent systems [1]. Conceptual shift from notion of single agent to multi-agent systems is essential paradigm to understand how two approaches differentiate. The most important concepts of both

paradigms are summarized in Table 1. In principle, an agent-based system might be conceptualized in terms of agents, but implemented without any software structures corresponding to agents at all. Therefore we can implement agent-based systems by enhancing the native functionality of programming languages and communication protocols if they are designed in terms of agents. For example, Java based multi-agent systems can be built upon the serialization, reflection, weak mobility (in case of mobile agents where code is downloaded and executed) and inter-object communication mechanisms.

BACKGROUND AND LITERATURE OVERVIEW

In literature two origins of the term 'agent' can be tracked down – the first one from the period of pioneers of agent research and the second one from the following wave of new researchers. In Bradshaw [2] roots of software agents are placed back in the 1950s when Selfridge and McCarthy proposed an idea of a goal-driven system, a 'soft robot' living and performing in the computer's world. The idea of software agents by early researchers has been further diversely researched and developed by new researchers in Distributed Artificial Intelligence (DAI), robotics, artificial life, distributed computing, intelligent and adaptive interfaces, information retrieval, knowledge acquisition and many other fields. Nwana [3] traces agents back to Hewitt's Actor model in the late 1970s. He proposed the concept of an agent he termed 'actor' and which had some characteristics of 'modern agents'.

Nwana split agent research into two main strands: the first beginning about 1977 with roots mainly in DAI and the second around 1990s studying much broader range of agent types. Strand 1 has considerably contributed to understanding of interaction, communication, coordination, cooperation and negotiation between agents. Basic aim of the agent supporting society in the nineties was to demonstrate the benefits of transforming standalone expert systems into a group of cooperating agents.

Literature overview

Started in the early eighties the research on software agents and multi-agent systems evolved into one of the most prevalent research areas in general computing. With the proliferation of the variety of agent types there has been an explosion in the use (and occasionally misuse) of the term 'agent' without a corresponding consensus on its meaning. Various classification schemes and taxonomies have been proposed. For more details, refer to Bradshaw's research [2]. Agents have been considered from AI and general computing perspectives and consequently the definition of agents has been approached at various levels of differing styles of argumentation [4]. Traditionally, research into systems composed of multiple agents was carried out under the banner of Distributed Artificial Intelligence (DAI), which has been divided into Distributed Problem Solving (DPS) and Multi-Agent Systems (MAS).

DPS is focused on solving a particular problem by a group of logically distinct components or modules (not necessarily agents), which cooperate in dividing and sharing knowledge about the problem and its developing solution [5]. Agents in typical DSP systems are often assumed to be benevolent i.e. they are explicitly designed to cooperatively achieve a common goal. The main concerns in DSP are how to divide a problem into smaller tasks for distribution among agents and how to effectively synthesize a problem solution from partial results.

In contrast, research in MAS is concerned with the societies of possibly pre-existing autonomous agents (i.e., agents with the type of properties listed in section about notions of agency) aiming at solving a given problem [6] but generally not sharing the common goal. Agents in MAS are usually considered self-interested, competitive or non-cooperative and may exhibit antagonistic behavior. They are often designed by different individuals or organizations in order to represent their interests, thus one agent's interests may conflict with interests of others. Despite the possibility of conflicting interests agents in MAS have to cooperate to achieve their goals. The main issues in multi-agent research are

how agents can cooperate, recognize and resolve conflicts negotiate or compromise in conflicting situations [7].

More recently, the term "multi-agent systems" has more general meaning and is referring to all types of systems composed of multiple (semi-)autonomous components [5]. Coordination is central to MASs, for without it, a group of agents quickly degenerates into a collection of individuals with a chaotic behavior. Coordination has been studied by researchers in diverse disciplines in the social sciences, including organization theory, political science, social psychology, anthropology, law and sociology, which produced many coordination mechanisms like Contract-Net Protocol, blackboard systems, market mechanisms or formal dialogue games. The most renowned coordination technique for task and resource allocation among agents and determining organizational structure is the Contract-Net Protocol [8]. Agents can use negotiation for solving conflict and hence for coordinating their activities. Other researchers place a stronger definition on negotiation arguing that in order to negotiate effectively, agents must reason about beliefs, desires and intentions of other agents [9]. In order to achieve this coordination, the agents might have to negotiate, however, they must interact and exchange information; therefore they need to communicate. An Agent Communication Language (ACL) provides agents with a means to exchange information and knowledge [9]. Most ACLs are based on speech act theory. In multi-agent systems, the possible solutions to the communication problem range from those involving no communication, those involving ad hoc ACLs to those involving standard ACLs like KQML [10] and FIPA ACL [6].

For agents to realize their potential as a software engineering paradigm, it was necessary to develop software engineering techniques that are specifically tailored to them. In order to support MAS requirements such as coordination, negotiation, and communication, several architectures have been reported in the literature, among them are Gaia, RETSINA, Multiagent Systems Engineering (MaSE),

AUML, Tropos, Prometheus, DECAF and others [12].

To date, the main areas in which agent-based applications have been reported are as follows: manufacturing, process control, telecommunication systems, air traffic control, traffic and transportation management, information filtering and gathering, electronic commerce, business process management, human capital management, skills management, (mobile) workforce management, defense, education,

e-learning, entertainment and medical care [6].

THE CONCEPT OF AN AGENT

Many definitions for the software agents, usually referred to as agents, have been given, but the most broadly accepted definition sees agents as autonomous, problem-solving computational entities capable of effective operation in dynamic and open environments [6]. Agents are often deployed in environments in which they interact, and maybe cooperate, with other agents (including both people and software) that have possibly conflicting aims.

Notions of agency

One of possibilities to classify agents is by different notions of agency. The notions of agency represent sets of properties systems should have in order to be recognized as agents. In the weak notion of agency defined by Wooldridge and Jennings [13] an agent is a system that enjoys the following properties:

Autonomy - agents can operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state; they have their own will.

Social ability – agents are able interact with each other with the use of some kind of agent-communication language.

Reactivity - agents perceive their environment and respond in a timely fashion to changes in the environment.

Pro-activeness - agents do not simply respond to stimuli in their environment, they are able to take the initiative.

In the strong notion of agency the weak notions of agency are preserved. In addition some characteristics of mentalistic notions, such as knowledge, belief, intention, and obligation are considered. Some other properties discussed in the context of strong agency are [1]:

Mobility – mobile agents have the ability to move around in the environment in a self-directed way.

Learning – agents should have the ability to learn while acting and reacting in its environment.

Temporal continuity – agents perform actions through a continuous running process.

Veracity – agents are assumed to be truthful.

Benevolence - agents do not have conflicting goals and they do what they are told to do.

Rationality – agents will perform in an optimal matter to achieve goals.

The relationship between agents and objects

Agents are often related to objects, where the letter are defined as computational entities that encapsulate some state, are able to perform actions (or methods) on this state, and communicate by passing a message. Obviously there are some similarities, yet there are also significant differences between agents and objects. First, agents are active autonomous entities capable of exercising choice over their actions and interactions; they cannot be directly invoked as objects, which are considered static entities. Second, they are capable of flexible, reactive, pro-active and social behavior. Third, a multi-agent system is intrinsically multi-threaded; each agent is assumed to have at least one thread of control, while object get a thread control from the calling object. Moreover, agents also support

structures for representing mental components, i.e. beliefs and commitments, and are capable of high-level interaction, using agent-communication languages between agents based on the "speech act" theory as opposed to ad-hoc messages frequently used between objects. Nevertheless, agents may be constructed using object-oriented technology.

Agent architectures

Agent architectures are the fundamental engines underlying the autonomous components that support effective behavior in real-world, dynamic and open environments. Wooldridge and Jennings [13] indicate that agent architectures can be viewed as software engineering models of agents. Agent architecture describes how agents perceive the environment through sensors and act upon that environment through effectors.

Four basic types of architectures have been identified [14]:

Deliberative or logic based (symbolic) architectures

Reactive (behavioral) architectures

Belief-Desire-Intention (BDI) architectures

Hybrid (layered) architectures

Deliberative architectures adopt the traditional AI (called symbolic AI) approach to designing intelligent systems by viewing them as a type of knowledge-based system. A deliberate agent is one which contains an explicitly represented, symbolic model of the world, and in which decisions including about what actions to perform, are made by some kind of logical reasoning [13].

Reactive architectures aim to develop and combine individual behaviors of reactive agents situated in some environment [14]. Reactive agents have a very simple representation of the world and operate in a simple stimulus-response fashion. Intelligent behavior emerges from the interaction of various simpler behaviors as well as from the interaction between an agent and its environment. The main disadvantage

of this architecture relates to the fact that agents do not employ models of their environment, therefore they need a great deal of local information to determine an acceptable action.

Belief-Desire-Intention architectures represent an agent in the terms of its beliefs, desires (goals) and intentions [15]. The basic components of a BDI agent are data structures (that represent beliefs, desires and intentions) and functions for representing them and reason about them. The agent's *beliefs* correspond to the information the agent has about its environment, its *desires* to the available actions (allocated tasks) and intentions to those desires that agent wants to achieve (agent's current focus).

Hybrid architectures combine the deliberative and reactive approaches. An agent in this architecture consists of a deliberative component and a reactive component [16]. Subsystems in agent's deliberative component develop plans and make decisions using symbolic reasoning, while subsystems in reactive component can react quickly to events without complex reasoning. A popular approach to designing hybrid agents is the use of layered architectures. Various subsystems of the architecture are arranged into a hierarchy of interacting layers each of which is reasoning about the environment at different levels of abstraction. Information and control can flow either horizontally or vertically. The input can be processed from each layer individually (*horizontal flow*), or from each layer after the other with the final layer forming the output (*vertical flow*).

Agent infrastructure

Agent infrastructure is concerned with developmental and operational support for agent systems. In the last few years, several new technologies have emerged that are aimed specifically at the ad-hoc networking that is central to the support of significant agent-based systems. These include Jini, UPnP and Salutation, for example, which define discovery and registration protocols that allow for dynamic discovery. Similarly, markup languages such as XML and RDF(S), along with standardized ontologies, provide a means for resource description and manipulation of this data at a semantic level.

Agent infrastructure also provides management functionality through such mechanisms as Jini leasing, which controls access to registry services, communication support from underlying transport mechanisms to robust protocols for information exchange, and security support to ensure that agents are properly authenticated and suitably authorized to perform their required actions [6].

Agent typology

Several classification schemes or taxonomies for classification of software agents have been proposed in the agent research community. The most acknowledged ones are Gilbert's scope of intelligent agents [17], typology based on Nwana's primary attributes dimension [3] and Franklin and Graesser's agent taxonomy [18]. Gilbert and his co-workers positioned intelligent agents along three dimensions – agency, intelligence and mobility, while Nwana proposed a typology of agents that identifies other dimensions of classification. Agents may thus be classified according to:

Mobility, as static or mobile

Presence of a symbolic reasoning model, as deliberative or reactive

Exhibition of primary attributes, such as autonomy, cooperation and learning

Roles, as Information or Internet

Hybrid combined approaches in a single agent

Secondary attributes, such as versatility, benevolence, veracity, trustworthiness, temporal continuity, mentalistic and emotional qualities.

Based on proposed typology, Nwana identified seven categories of agents: collaborative agents, interface agents, mobile agents, information/Internet agents, reactive agents, hybrid agents, and smart agents. According to Franklin and Graesser, autonomous agents are divided into biological agents, robotic agents and computational agents, the last ones being further divided into software agents and AI agents. At last, software agents are divided into task-specific agents, entertainment agents and viruses.

MULTI-AGENT SYSTEMS

Although, in many cases, agents can act separately to solve a particular problem, it often happens that a complete system made of several different agents has to be designed to cope with a complex problem involving either distributed data, knowledge, or control.

Agents are often deployed in environments in which they interact, and maybe cooperate, with other agents – including both people and software – that have possibly conflicting aims. Such environments are known as multi-agent systems. A multi-agent system (MAS) is a collection of possibly pre-existing autonomous agents that communicate between them to coordinate their activities in order to be able to solve collectively a problem that could not be tackled by any agent individually [19]. Multi-agent systems are based on the idea that a cooperative working environment comprising synergistic software components can cope with problems which are hard to solve using the traditional centralized approach to computation.

In order for a MAS to solve common problems coherently, the agents must communicate amongst themselves, coordinate their activities and negotiate once they find themselves in conflict. Conflicts can result from simple limited resource contention to more complex issue-based computations where the agents disagree because of discrepancies between their domains of expertise. Coordination is required to determine organizational structure amongst a group of agents and for task and resource allocation, while negotiation is required for the detection and resolution of conflicts.

Characteristics of multi-agent systems

The fundamental aspects that characterize a multi-agent system and distinguish it from a single-agent system can be observed along several dimensions [20]:

Agent design: Agent heterogeneity can affect all functional aspects of an agent from perception to

decision making. In single agent systems this issue is nonexistent.

Environment: Agents inhabit either static (time invariant) or dynamic (non-stationary) environments, but in a MAS, the presence of multiple agents makes the environment appear dynamic from the agent's point of view.

Perception: The collective information that reaches agents' sensors is typically distributed spatially (at different locations) or even semantically (requiring different interpretations).

Control: The control in a MAS is typically decentralized.

Knowledge: In a MAS the levels of knowledge of each agent about the current world state can differ substantially. In general, in a MAS each agent must also consider the knowledge of each other agent in its decision making. A crucial concept here is that of common knowledge, according to which every agent knows a fact, every agent knows that every other agent knows this fact, and so on.

Communication: Interaction is often associated with some form of communication. Typically we view communication in a MAS as a two-way process, where all agents can potentially be senders and receivers of messages. Communication can be used for coordination among cooperative agents or for negotiation among self-interested agents. Communication can be present in different forms – as argumentation, negotiation or persuasion. Effective communication among agents requires shared knowledge of syntax, shared understanding of semantics and pragmatics and some means of exchanging messages to communicate.

Characteristics of multi-agent systems environments

Environments for multi-agent systems have to provide computational infrastructure for agents, enable protocols for communication end enable interactions between agents so that they may share resources and coordinate their activities. The infrastructure has to provide shared resources for agents, communication and interaction protocols and transportation methods for mobile agents. MAS

environments are usually open, distributed and may be based on standards. Inhabitants are autonomous agents that communicate with the environment and may be cooperative or selfish. These environments have to let agents operate effectively and let them interact productively. Table 2 summarizes the classification of the agent environments based on their properties introduced by Russell and Norvig [21].

AGENT TECHNOLOGIES

Coordination, cooperation and communication mechanisms

The exchange of knowledge and information between the agents in MAS is important for solving problems efficiently and coherently. An agent in MAS has to be able to communicate with other agents and with the environment in order to coordinate activities, to negotiate in the case of a conflict and to be able to allocate tasks and resources. Agent communication languages enable agents to collaborate with each other by providing them with the means of exchanging information and knowledge [11]. Agent communication languages remain just syntax without a shared common ontology containing the terms used in agent communication and the knowledge (e.g. definitions, attributes, relationships between terms and constraints) associated with them [22].

Coordination is considered as an essential aspect of MAS. It prevents chaos when conflicts occur between agents and it places global constraints for the behaviors of agents. It is necessary because agents in MAS have different and limited capabilities or expertise and their activities might be interdependent [23]. Even if agents' activities are independent, one agent can discover information that is useful for another agent, so that coordination also enables efficiency. Nwana et al. [24] indicated that coordination may require cooperation (although coordination can also occur without cooperation) but cooperation among agents does not necessarily result in coordination. Additionally, communication

among agents may be required for coordination but agents can also be coordinated without communication, if agents possess the models of each other's behaviors.

Agent communication languages

An agent communication language (ACL) provides agents with a means to exchange information and knowledge. Over the years other means and approaches have been used for exchanging knowledge and information between applications, among them are Remote Procedure Call (RPC) or Remote Method Invocation (RMI), CORBA and Object Request Brokers (ORB's) [2]. ACLs like KQML or FIPA ACL can handle propositions, rules and actions instead of simple objects and the ACL message describes a desired state in a declarative language, rather than a procedure or method.

KQML is a high level, message-oriented communication language and protocol for information exchange. It is independent of the transport mechanism ((TCP/IP, SMTP, IIOP, etc.), independent of the content language (KIF, SQL, STEP, Prolog, etc.) and independent of the ontology assumed by the content. The KQML language is divided into three layers:

Content layer: Bears the actual content of the message.

Message layer: Encodes a message that one application would like to transmit to another. The message layer forms the core of the KQML language, and determines the kinds of interactions one can have with a KQML-speaking agent.

Communication layer: Encodes a set of features to the message which describe the lower level communication parameters, such as the identity of the sender and recipient, and a unique identifier associated with the communication.

The FIPA Agent Communication Language (*FIPA ACL*), like KQML, is based on speech act theory: messages are actions, or communicative acts, as they are intended to perform some action by virtue of being sent. The specification consists of a set of message types and the description of their pragmatics,

that is, the effects on the mental attitudes of the sender and receiver agents. Every communicative act is described with both a narrative form and a formal semantics based on modal logic.

Learning in agents

Learning ability is a crucial feature of intelligent agents, especially when faced with a multi-agent environment. Agents operate in a dynamic environment, making it impossible to statically determine an agent's optimal behavior in advance. Agents have to learn from and adapt to their environment. Multi-agent learning and adaptation, that is, the ability of agents to learn how to cooperate and compete, becomes crucial. Applications of learning agent technology have been especially successful in the areas of personalization and information retrieval, and promising results have been achieved in the areas of robotics and telecommunications. Over the years, learning and adaptation has occupied researchers from disciplines such as artificial intelligence, machine learning, information retrieval and Human-Computer Interaction (HCI), and in the agent domain, work has also concentrated around other areas including adaptive user interfaces, user profiling, and personalization techniques [5].

Single agent learning and multi-agent learning: To date, most learning algorithms have been developed from a single-agent perspective. Single-agent or isolated learning focuses on how one agent improves its individual skills, irrespective of the domain in which it is embedded. Single-agent learning might not always yield an optimal performance in multi-agent environments and there may be domains where coordinated multi-agent learning is a more natural metaphor and improves the effectiveness. On the contrary, multi-agent (or interactive) learning relies on the presence of multiple agents and their interaction. The concept of interactive learning itself can be applied in two different ways. In the stronger and more specific meaning, interactive or multi-agent learning refers only to situations in which several agents learn how to pursue a common learning goal. In the weaker and less specific meaning, it additionally refers to situations in which an agent pursues its own learning goal, but is

affected in its learning by other agents.

On-line and off-line learning methods: On-line (or incremental) learning algorithms, such as ant colony optimization algorithms and nature-inspired paradigms such as artificial immune systems, have been used to compute new hypotheses incrementally as soon as a new training example becomes available. On the other hand, off-line learning methods induce a hypothesis from a set of training examples presented to the algorithm at a single time point. Apparently, on-line algorithms are better suited for multi-agent systems where agents need to update their knowledge constantly. However, it is desirable to be able to use the large and powerful class of off-line learning algorithms as well. In order to do this, an agent needs to collect a set of training examples and then decide at some time point to compute (or re-compute) a hypothesis.

Principles from biology

In this chapter we try to compare the typical biological examples that inspired scientists in development of multi-agent theory, where they were often presented as reactive agents. One of the first papers in this field written by Dorigo et al. [25] introduces the colonies of cooperating agents behaving like a colony of ants. Later this analogy with the way ant colonies function has suggested the definition of a new computational paradigm, which is called Ant Colony Optimization or Ant Systems. In this approach the search for solutions is distributed over agents using very simple basic capabilities which, to some extent, mimic the behavior of real ants. The basic idea was taken from ethologists who found out that almost blind animals like ants could manage to establish the shortest route from their colony to specific destinations. They found out that ants can communicate information among individuals regarding paths, using pheromones. The way pheromones work is analogous to the way hormones in the body send specific chemical signals from one set of cells to another, causing them to perform a certain

action.

Another characteristic of agent based systems is a process of self-organization which can also be found in nature. It is a process in which pattern at the global level of a system emerges solely from multiple interactions among the lower level components of the systems. There are five basic features of complex self-organizing systems that can be directly applied to agent systems:

Large number of subjects

Large number of interactions

Simple interaction rules

Decentralized organization

Emergent behavior

Although current multi-agent systems cannot match the complexity level of complex biological systems, we can see the obvious similarities.

APPLICATIONS OF AGENT SYSTEMS

Intelligent Agent systems have been applied in many biomedical fields worldwide. Although most of the agent based systems still exist only in the experimental environments, there are also some systems that are already used in everyday life. We should also be aware of the fact that these new agent-based systems should take into account rapidly changing national and international laws and regulations concerning the privacy of medical data and the security of the transaction of patient information between agents which can also be the difficulty when deploying agent-based system to the medical environment.

We will divide the applications in the following groups:

Patient monitoring

Knowledge management agents

Distance learning systems

Community care

Organ and tissue transplant management

Patient and staff scheduling

Patient monitoring

The Guardian system proposed by Hayes-Roth [26] is possibly not just the patient monitoring multi-agent system, but also one of the first applications of the agent-based systems in biomedical field. It is intended to help in the patient monitoring process in Surgical Intensive Care Units (SICU). The system tries to distribute the SICU patient monitoring among a number of agents, of three different types:

Perception/action agents – interface between Guardian system and the world

Reasoning agents – organization of decision making

Control agents – top-level control of the system

Another similar system was proposed by Huang et al. [27] where they try to model many individuals who contribute in patient management and care process. For example, a general practitioner may suspect that a patient has breast cancer, but this suspicion cannot be confirmed or rejected without the assistance of a hospital specialist. If the specialist confirms the hypothesis, then a care procedure must be devised for treating the patient, involving the resources of other individuals. The system allows a natural representation of this process, with agents mapped onto the individuals [28].

Knowledge management agents

Agent based systems for knowledge management in biomedical field are used to acquire, create, organize, share and store biomedical knowledge. Most of systems for biomedical knowledge

engineering use World Wide Web resources and are built upon information that is acquired from the internet. In many cases such systems are used as medical decision support systems [29]. One of such systems is the Multi-Agent Retrieval Vagabond on Information Networks (MARVIN) [30], developed at the Health on the Net Foundation and the Swiss Institute of Bioinformatics, where they try to index medical documents on the World Wide Web. MARVIN is intended to filter documents in eight languages and provide better help by exploiting a multilingual index of medical papers. In the later stages of development it was used to feed the medical and health oriented search engine called MedHunt. Another similar system, called Personalised Information Retrieval Agent (PIRA), was developed by Lobato and Shankararaman [31], where an information agent, based on a user profile, can proactively perform the role of locating, assessing, retrieving, filtering and presenting information from many distributed sources on a periodic basis. Another medical diagnosis system that combines the advantages of multi-agent system technologies and neural networks in order to realize a highly reliable, adaptive, scalable, flexible, and robust medical diagnosis was presented by Klüver et al. [32]. The system works as a hierarchically organized structure of agents that collaborate in a search for the viable medical diagnosis. Higher level agents cover broader field of diseases, while on the lover levels we can find more specialized experts.

Distant learning systems

Usually we use distant learning systems in the education process at the medical or health care educational institutions, but they can be used as the tutoring and learning applications for the medical staff as well. One of the well known systems is called Adele (Agent for Distance Education - Light Edition) and was developed at the Center for Advanced Research in Technology for Education [33]. Adele is an animated pedagogical agent. Such agents have animated personas that respond to user actions. In addition, they have enough understanding of the learning context and subject matter that

they are able to perform useful roles in learning scenarios. The system is designed in a way that supports students working through problem-solving exercises that are integrated into instructional materials delivered over the World Wide Web.

Community care

Special interest groups for the application of multi-agent systems are the senior and disabled citizens. The agent based technology can significantly improve the quality of live in such communities. Multi-agent systems can provide aid of carrying out the daily activities, link with the outside world, and access to the information and communication with the family to the senior and disabled people. Actually such systems share a lot of functions with the similar systems for monitoring patients, described in Patient monitoring section. These kinds of tools may be used to facilitate the health care and social interaction of senior citizens, and may delay their institutionalization by prolonging their period of independence [34]. A European project called TeleCARE is an example of the agent-based system that aims to offer assistance and support to the elderly people employing tele-supervision and tele-assistance [35].

Organ and tissue transplant management

Despite significant advances in the surgery process, the coordination of the preliminary activities involved in an organ transplant operation is still a very challenging and complex process. In organ transplant management (OTM) systems, agents take care of the coordination process that is needed to find the most appropriate recipient of the organ that is currently available somewhere in the region that is covered by the system. An example of OTM system was described in [36], where authors propose agent-mediated combination of tissue and organ transplant management system.

Patient and staff scheduling

Patient and staff scheduling in medical environments is a highly complex task. The problem is

very old, but due to the complexity the solutions have been proposed also using agent oriented approaches. Decker and Li proposed one of the first multi-agent systems for patient scheduling [37]. Much more advanced system for scheduling the patients inside the hospital environment was proposed by Paulussen et al. [38], where they introduce a multi-agent based distributed approach to patient scheduling under variable pathways and stochastic process durations.

Another multi-agent system was developed to solve nurse rostering problem. The technique involves a hybrid approach that devolves responsibility for different aspects of the problem to a heuristic component and a constraint solver. In the pre-processing stage, the staff to be rostered is treated as semiautonomous agents, each with individual responsibility for their initial assignment, and communicating with a global constraint satisfaction agent [39].

The approach which addresses both patient and staff appointment scheduling was presented in [40]. The presented multi-agent based simulation system includes optimization of patient and nurse scheduling in ambulatory health care environments. Additional to that the paper presents time-series forecasting in the patient scheduling domain.

More general approach was presented in [41] where authors introduced an agent based framework for medical appointment scheduling.

FUTURE OF AGENT SYSTEMS

A lot of changes in the software agents' field need to be done to change the current situation where agents exist mainly in academic and commercial laboratories, but are not widely available in real world applications. This move is likely to happen over the next few years, but there are still some issues that limit the availability of agent systems for the end users. In particular, there is still a lack of web standards that would enable structural and semantic information description. Over the next few years

Web services standards and standards for Semantic webs will play an important role in the agent standardization efforts. A further field where we lack support is standardization in the knowledge management software, where there are many solutions of storing knowledge, but everyone seems to be doing it in some specific way.

Another open issue for the future is the validity of such complex multi-agent systems that we started to build. The way to solve this problem lies in development of systems that would exhibit online adaptiveness rather than a priori proof validity [42]. This is strongly connected with the formalization of agent's ability to adapt and learn behaviors which will be another difficult task to solve.

Focusing on medical applications of agent systems there is currently a lot of research going on in the fields of communication and co-operation between distributed intelligent agents. Aim of the research in the mentioned directions is to automate agents which can manage information about patients and autonomously react and interact with other agents within medical environment. We should also take into account the rapidly changing national and international laws and regulations concerning the privacy of medical data in patient information transactions. Therefore it is obvious that in the foreseeable future, there will be a demand for closed multi-agent systems, mainly because of the security concerns that arise from open systems. Thorough reliability testing for agents, formal methods for open agent systems, trust techniques for coping with malicious agents and reputation mechanisms have to be developed to ensure trust in adopting agent technology.

BIBLIOGRAPHY

1. M. Wooldridge and P. Ciancarini, Agent-Oriented software engineering: The state of the art, in P. Ciancarini and M. Wooldridge, (eds.), *Agent-Oriented Software Engineering*, vol. 1957 of LNCS, Springer, pp. 1 – 28, 2001.

2. J. Bradshaw, An Introduction to Software Agents, *Software Agents*, J. Bradshaw (ed.), American Association for Artificial Intelligence/MIT Press, 1997.

3. H. S. Nwana, Software Agents: An Overview, *Knowledge Engineering Review*. 11(3): 1-40, 1996.

4. K. D. Reilly, 2001. Agents and lightweight use of logic: combined simulation, logic and neural agent nodes, *Proceedings of the 2000 Huntsville Simulation Conference*. International Society for Computer Simulation, San Diego, CA, 2000.

5. N. R. Jennings, K. Sycara and M. Wooldridge, A roadmap of agent research and development, *Autonomous Agents and Multi-Agent Systems*, 1(1):7–38, 1998.

6. M. Luck, P. McBurney and C. Preist, Agent Technology: Enabling Next Generation Computing (A Roadmap for Agent Based Computing), *AgentLink*, 2003.

7. M. Wooldridge, Agent-based computing, *Interoperable Communication Networks*, 1(1): 71-97, 1998.

8. R. Davis and R. Smith, Negotiation as a Metaphor for Distributed Problem Solving, *Artificial Intelligence*, 20: 63-109, 1983.

9. A. Rao and M. Georgeff, BDI Agents: From Theory to Practice, Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, USA, (June 1995).

10. Y. Labrou and T. Finin, A, Semantics Approach for KQML—A General-Purpose Communication Language for Software Agents, *Proceedings of the Third International Conference on Information and Knowledge Management*, New York: Association of Computing Machinery, 447-455, 1994.

11. Y. Labrou, T. Finin and Y. Peng, Agent Communication Languages: The Current

Landscape, *IEEE Intelligent Systems*, 1999.

12. S. Park and V. Sugumaran, Designing multi-agent systems: a framework and application, *Expert Syst. Appl.*, 28(2): 259-271, 2005.
13. M. J. Wooldridge and N. R. Jennings, Intelligent agents: Theory and practice, *The Knowledge Engineering Review*, 10(2):115–152, 1995.
14. M. Wooldridge, *Intelligent Agents*, The MIT Press, 1999
15. A. S. Rao and M. P. Georgeff, BDI Agents: From Theory to Practice, *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, San Francisco, USA, 1995.
16. H. Helin, Agent Architectures & Languages. [Online]. http://www.cs.helsinki.fi/u/hhelin/opetus/oat/ [2005, 15 March]
17. D. Gilbert, M. Aparicio, B. Atkinson, S. Brady, J. Ciccarino, B. Grosof, P. O'Connor, D. Osisek, S. Pritko, R. Spagna and L. Wilson, IBM Intelligent Agent Strategy, *IBM Corporation*, 1995.
18. S. Franklin and A. Graesser, Is It an Agent or Just a Program? A Taxonomy for Autonomous Agents, *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*. New York: Springer-Verlag, 1996.
19. M. Wooldridge, *An introduction to Multiagent systems*. John Wiley, Chichester, 2002.
20. N. Vlassis, *A Concise Introduction to Multiagent Systems and Distributed AI*, Introductory Text, University of Amsterdam, 2003.
21. S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2/E, Prentice Hall, 2003.
22. H. Nwana and M. Wooldridge, Software Agent Technologies. *BT Technology Journal*

14(4): 68-78, 1996.

23. K. Ciesielski, Towards the Adaptive Organization: Formation and Conservative Reconfiguration of Agents Coalitions, *Proceedings of Autonomous Intelligent Systems: Agents and Data Mining: International Workshop, AIS-ADM 2005*, 79-92, 2005.

24. H. Nwana, L. Lee and N. Jennings, Coordination in Software Agent Systems. *BT Technology Journal* 14(4): 79-88, 1996.

25. M. Dorigo, V. Maniezzo and A. Colorni, The Ant System: Optimization by a colony of cooperating agents, *IEEE Transactions on Systems, Man, and Cybernetics*–Part B, 26(1): 1-13, 1996.

26. B. Hayes-Roth, M. Hewett, R. Washington, R. Hewett and A. Seiver, Distributing intelligence within an individual. In: L. Gasser, M. N. Huhns (Eds.) *Distributed AI*, Volume II, 385-412. Morgan Kaufmann, 1990.

27. J. Huang, N. R. Jennings and J. Fox, An agent-based approach to health care management, *Int. Journal of Applied Artificial Intelligence*, 9(4): 401-420, 1995.

28. N. Jennings and M. Wooldridge, Applications of Intelligent Agents, *Agent Technology: Foundations, Applications, and Markets*, N. R. Jennings and M. Wooldridge (eds.), 3-28, 1998.

29. L. Lhotska, O. Stepankova, Agent architecture for smart adaptive systems, *Transactions of the Institute of Measurement and Control*, 26(3): 245-260, 2004.

30. O. Baujard, V. Baujard, S. Aurel, C. Boyer and R. D. Appel, MARVIN, a multi-agent softbot to retrieve multilingual medical information on the Web. *Medical Informatics* 23 (3): 187-191, 1998.

31. E. Lobato and V. Shankararaman, PIRA: A Personalised Information Retrieval Agent, *Proc.*

of IASTED International Conference on Artificial, Intelligence and Soft Computing, 1999.

32. J. Klüver, C. Stoica, M. Ulieru and R. Unland, Medical diagnosis by interacting neural agents, *Journal of Soft Computing - A Fusion of Foundations, Methodologies and Applications*, Heidelberg, Springer, 2004.

33. E. Shaw, W. L. Johnson and R. Ganeshan, Pedagogical Agents on the Web, *Proceedings of the 3rd International Conference on Autonomous Agents*, 283-290, 1999.

34. J. L. Nealon and A. Moreno, Agent-Based Applications in Health Care, *Applications of Software Agent Technology in the Health Care Domain* (eds Nealon, J.L and Moreno, A.), Whitestein Series in Software Agent Technologies, Birkhäuser Verlag, Basel, 3-18, 2003.

35. L. M. Camarinha-Matos, Tele-Care and Collaborative Virtual Communities in Elderly Care, *Proceedings of the 1st International Workshop on Tele-Care and Collaborative Virtual Communities in Elderly Care*, *TELECARE 2004*, INSTICC Press, 2004.

36. J. Vázquez-Salceda, U. Cortés and J. Padget, Integrating the Organ and Tissue Allocation Processes through an Agent-Mediated Electronic Institution, *Proceedings of the 5th Catalonian Conference on AI, CCIA 2002*, Castellón, Spain, 309-321, 2002.

37. K. Decker and J. Li, Coordinated Hospital Patient Scheduling, *Proceedings of the 3rd International Conference on Multi Agent Systems (ICMAS '98)*, 104-111, 1998.

38. T.O. Paulussen, N.R. Jennings, K.S. Decker and A. Heinzl: Distributed Patient Scheduling in Hospitals, *Proceedings of IJCAI*, 1224-1232, 2003.

39. G. Winstanley, A hybrid AI approach to staff scheduling, *Proceedings of the 22nd Annual International Conference of the BCS Special Interest Group on Artificial Intelligence: SGES 2002*, Cambridge University, 2002.

40. G. Stiglic, P. Kokol, Patient and Staff Scheduling Multi-Agent System, *Proceedings of the*

3rd International Conference on Computational Cybernetics, 25-28, 2005.

41. M. Hannebauer and S. Müller, Distributed Constraint Optimization for Medical Appointment Scheduling. *In Proceedings of the 5th International Conference on Autonomous Agents (AGENTS-2001)*, 139-140, 2001.

42. D. Servat and A. Drogoul, Combining amorphous computing and reactive agent-based systems: a paradigm for pervasive intelligence?, *Proceedings of the first international joint conference on Autonomous agents and multiagent systems (AAMAS '02)*, 441-448, 2002.

TABLES

Table1: The most important concepts of agent and MAS approaches

Table 1

Agent	**MAS**
Categories	Social organization
Architectures	Coordination, cooperation and interaction
Knowledge representation and management	Planning and learning
Knowledge acquisition and inference	Forms of communication
	Mental models

Table2: Classification of multi-agent system environments

Table 2

Environment	**Description**
Accessible / Inaccessible	Agents can obtain complete, accurate, up-to-date information about the environment's state in an accessible environment. Most complex environments are inaccessible.
Deterministic / Non-deterministic	Any action of the agent has a single guaranteed effect in a deterministic environment as opposed to some uncertainty about the resulting state after an action is performed in a non-deterministic environment.
Episodic / Non-episodic	The performance of an agent depends on a number of discrete episodes in an episodic environment. There is no link between the performances of an agent in different scenarios.
Static / Dynamic	A static environment can be changed only by the performance of the actions of the agent while a dynamic environment has other processes operating in it, which are not under the control of the agent.
Discrete / Continuous	There are a fixed, finite number of actions and percepts in a discrete environment.