Preprints of the 18th IFAC World Congress
Milano (Italy) August 28 - September 2, 2011

Designing Adaptive Multi-agent Systems by Agent-Oriented Modelling

Kuldar Taveter*, Kristi Kirikal*

**Department of Informatics, Tallinn University of Technology, Tallinn, Estonia (e-mails: kuldar.taveter@ttu.ee, kristi.kirikal@gmail.com)*

Abstract: The challenges in creating software for modern complex and distributed computing environments are time-sensitivity, uncertainty, unpredictability, and openness. It is a problem how to design systems that work effectively in the modern environment, where computing is pervasive, people interact with technology existing in a variety of networks, and under a range of policies and constraints imposed by the institutions and social structures that we live in. The key concepts we use for designing open, adaptive, distributed, and self-managing systems are agents and sociotechnical systems. An agent is suitable as a central modelling abstraction for representing distributed interconnected nodes of the modern world. A sociotechnical system encompasses a combination of people and computers, hardware and software, many of which can be viewed as agents. This article addresses the bottom-up development of sociotechnical systems as adaptive multi-agent systems. We claim that the methods for designing adaptive multi-agent systems should be borrowed from social sciences rather than from exact sciences. In particular, we propose a solution for developing flexible multi-agent systems in such a way that an environment does not have to be analysed in all its complexity.

Keywords: agents, modelling, sociotechnical system design, adaptive systems, social and behavioural sciences.

1. INTRODUCTION

A modern computing environment is characterized by a transition from client-server computing to *peer-to-peer* computing. Under the new paradigm, all the nodes of the network are equal and each node can function both as a server or client. Interactions between nodes rely on various services, such as directory services and communication services. We thus understand *peer-to-peer* computing more broadly than file sharing systems like Napster or VoIP applications like Skype. Intelligent agents and multi-agent systems are effectively rooted in the *peer-to-peer* paradigm. When we also consider humans participating in *peer-to-peer* networks, we are faced with a complex distributed environment for which modern software applications must be written. We need a way for conceptualizing and modelling distributed applications and their environments neutrally with respect to specific software platforms and architectures. Because, as was stated earlier, multi-agent systems are rooted in the new peer-to-peer paradigm instead of the traditional client-server paradigm, object-oriented analysis and design are *not* adequate for engineering of such systems.

The challenges in creating software for modern complex and distributed computing environments are described by Sterling & Taveter (2009). They are time-sensitivity, uncertainty, unpredictability, and openness. It is a problem how to design systems that work effectively in the modern environment, where computing is pervasive, people interact with technology existing in a variety of networks, and under a range of policies and constraints imposed by the institutions and social structures that we live in. The key concepts that Sterling & Taveter (2009) use for designing open, adaptive, distributed, and self-managing systems are agents and sociotechnical systems. An agent is suitable as a central modelling abstraction for representing distributed interconnected nodes of the modern world. A sociotechnical system encompasses a combination of people and computers, hardware and software.

The novelty of the approach presented in this article as compared with (Sterling & Taveter, 2009) is that it explains why the methods for designing adaptive sociotechnical systems should be borrowed from social sciences rather than from exact sciences and demonstrates how it can be done.

2. SOCIOTECHNICAL SYSTEMS

Sociotechnical systems are more complex than merely technical systems. Technical systems may be studied by using the methods of exact sciences. However, these methods are not applicable to sociotechnical systems because the characteristics of social systems should be considered when designing such systems. As Prigogine (1997) pointed out, the world is a complex system which develops in irreversible time. It is impossible to re-create the same situation in a social environment because social experiments are not conducted in a laboratory. Social experiments have impact on society and therefore initial conditions will also change. A social system can be viewed as having two kinds of statuses. These modes are “is” and “is not” or “agree” and “disagree”, depending on the situation. The action of choosing a status by an agent triggers some event. Agents are active components

of sociotechnical systems. Examples of agents are people, software agents, and robots. In addition, sociotechnical systems are *here-and-now* systems or run-time systems that cannot be reset and re-generated but should incorporate new functionality gradually at run-time. In other words, sociotechnical systems should be flexible to fit in a changing environment.

Popper states that no scientific predictor – whether a human scientist or a calculating machine – can possibly predict, by scientific methods, its own future (Popper, 1964). Luhmann (1995) claims that "everything that happens belongs to a system (or to many systems) and *always at the same time* to the *environment of other systems*". In contemporary complex systems, the boundaries between systems and their environments are not clear. Luhmann (1995) claims that "Complexity can help to clarify system/environment difference. Establishing and maintaining the difference between system and environment then becomes the problem, because for each system the environment is more complex than the system itself".

As describing the environment of a complex system poses a problem, system design and developing strategies need to suggest alternatives to describing environments in profound details. We propose a solution for developing flexible multi-agent systems in such a way that an environment does not have to be analysed in its entire complexity.

An environment does not have to be thoroughly analysed because an agent chooses its actions based on the information available at a given moment. If the information required is not available in a current situation, the agent will use the information that is available or will try again after a while. However, that will be another *here-and-now* situation.

Sociotechnical systems should follow patterns of social systems. In a society, each person is an autonomous agent. No one knows information about the society as a whole but everyone knows the information necessary to fulfil its objectives. Moreover, no one, not even the University's Rector, knows all the information about a reasonably large organization such as Tallinn University of Technology. However, society as a complex system works reasonably well, despite the fact that each member of a society knows only a very small part of the whole system. We are convinced that this approach is also applicable to engineering multi-agent systems.

Based on the preceding arguments, when designing sociotechnical systems, there is no need to describe their environments as accurately as possible (and as was previously indicated, it is not even possible). What matters is that each agent knows enough about its specific objectives and about the means for achieving them.

Our approach is rooted in agent-oriented modelling proposed by Sterling & Taveter (2009). However, instead of using agent-oriented modelling for just *top-down* development of multi-agent systems, as proposed by Sterling & Taveter (2009), we propose to apply agent-oriented modelling also to iterative *bottom-up* development of multi-agent systems. In the following section, we give an overview of agent-oriented modelling.

3. AGENT-ORIENTED MODELLING

Agent-oriented modelling as advocated by Sterling & Taveter (2009) presents a holistic approach for analysing and designing sociotechnical systems consisting of humans and technical components. Agent-oriented modelling proposes a set of canonical models. The models for analysing a problem domain describe the functional goals of a sociotechnical system (e.g., information system) to be designed, the quality goals describing how the functional goals should be achieved, the roles required for achieving the goals, and the domain entities capturing the knowledge to be represented within the system. Design models of agent-oriented modelling describe what human and man-made agents are required for achieving the goals, what private and shared information these agents possess and process, and how they interact and behave. Analysis models and design models are complemented by platform-specific design models that describe the implementation of the sociotechnical system on a particular software platform.

The types of models proposed by agent-oriented modelling are represented in Table I. In addition to representing for each model the abstraction layer (analysis, design, or platform-specific design), Table I maps each model to the vertical viewpoint aspect of interaction, information, or behaviour. Each cell in the table represents a specific viewpoint. We will next give an overview of agent-oriented models proceeding by viewpoints.

From the viewpoint of *behaviour analysis*, a *goal model* can be considered as a container of three components: goals, quality goals, and roles (Sterling & Taveter, 2009). A *goal* is a representation of a functional requirement of the sociotechnical system to be developed. A *quality goal*, as its name implies, is a non-functional or quality requirement of the system. Goals and quality goals can be further decomposed into smaller related subgoals and subquality goals. The hierarchical structure is to show that the subcomponent is an aspect of the top-level component. Goal models also determine roles that are capacities or positions that agents playing the roles need to contribute to achieving the goals. Roles are modelled in detail in the viewpoint of interaction analysis. The notation for representing goals and roles is shown in Table II. This notation is used in Section 4 in presenting agent-oriented models for the example case study. Goal models go hand in hand with *motivational scenarios* that describe in an informal and loose narrative manner how goals are to be achieved by agents enacting the corresponding roles (Sterling & Taveter, 2009).

From the viewpoint of *interaction analysis*, the properties of roles are expressed by role models. A *role model* describes the role in terms of the responsibilities and constraints pertaining to the agent(s) playing the role. *Organisation model* is a model that represents the relationships between the roles of the sociotechnical system, forming an organization (Sterling & Taveter, 2009).

From the viewpoint of *information analysis*, *domain model* represents the knowledge to be handled by the sociotechnical system. A domain model consists of domain entities and relationships between them. A domain entity is a modular unit of knowledge handled by a sociotechnical system (Sterling & Taveter, 2009).

From the viewpoint of *interaction design*, *agent models* transform the abstract constructs from the analysis stage, roles, to design constructs, *agent types*, which will be realized in the implementation process. The *acquaintance model* complements the agent models by outlining interaction pathways between the agents of the system. *Interaction models* represent interaction patterns between agents of the given types. They are based on responsibilities defined for the corresponding roles.

From the viewpoint of *information design*, the *knowledge model* describes the private and shared knowledge by agents of the multi-agent system to be designed. Finally, from the perspective of *behaviour design*, *scenarios* and *behaviour models* describe the behaviours of agents in the system.

As was stated at the end of Section 2, this article applies agent-oriented models to iterative bottom-up development of multi-agent systems.

TABLE I. THE MODEL TYPES OF AGENT-ORIENTED MODELLING

	Viewpoint aspect		
Abstraction layer	Interaction	Information	Behaviour
Analysis	Role models and organization model	Domain model	Goal models and motivational scenarios
Design	Agent models, acquaintance model, and interaction models	Knowledge model	Scenarios and behaviour models
Platform-specific design	Platform-specific design models		

TABLE II. NOTATION FOR MODELLING GOALS AND ROLES

Symbol	**Meaning**
	Goal
	Quality goal
	Role
	Relationship between goals
	Relationship between goals and quality goals

4. ITERATIVE DEVELOPMENT

In this section we propose an approach for iterative bottom-up development of multi-agent systems based on agent-oriented modelling. The rationale for this kind of development is that contemporary complex systems can have no agents who know all the information. It is not even necessary to have such agents. Instead, it is important for each agent to know its objectives and the means of achieving them and be situated in an environment necessary for achieving the objectives. In our view, multi-agent systems should be developed in iterative phases. A system in its any phase should include at least one agent who is aware of the goal which should be achieved by the system as a whole, i.e. the system's purpose. When an agent knows the system goal, a complex system environment does not have to be described in detail. Each agent knows its objectives and this is sufficient to achieve the system goal.

In our approach, agents belong to different abstraction levels. First-level agents are always used because they do the actual work, such as assembling cars or cell phones on a production line. Therefore agents of the first-level are created before agents of other levels. Figure 1 represents two first-level agents who know each other. If the goals to be achieved by these agents are straightforward, these agents can coordinate their activities just between themselves. The coordination may lie in passing the product that is being assembled from one agent to another in a timely manner. This kind of situation is depicted in Figure 1.

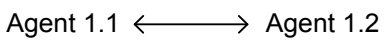

Figure 1. Acquaintance model for agents of the first level.

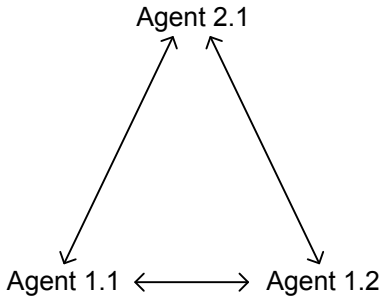

Figure 2. Acquaintance model for agents of the second level.

Let us suppose that the requirements of a sociotechnical system are represented in the form of a goal model of agent-oriented modelling. If new goals and/or roles are added to the goal tree, more agents may need to be created at the same level and/or at higher levels. In particular, agents of a higher level have to be created if agents of lower level(s) need more coordination to achieve their objectives. In Figure 2, an agent of the second level has been added who interacts with agents of the first level.

An agent of a higher level might not be aware of (all) the agents at the levels below it. This is exemplified by Figure 3, according to which an agent of the third level is aware of just one agent of the second level, who is probably the manager of the second-level agents.

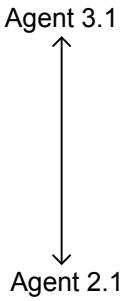

Figure 3. Acquaintance model for agents of the third level.

We illustrate our considerations by introducing an example from the problem domain of assembling cell phones by a sociotechnical automation system consisting of autonomous robots and humans. The robots are equipped with sensors and actuators and are capable of reasoning and interactions.

We describe the requirements for the sociotechnical system in the form of a goal model. Figure 4 shows that for achieving the purpose of the system – Assembling cell phone – several subgoals need to be achieved. Just one agent – an agent playing the Manager role – is aware of the system purpose: Assemble cell phone. For achieving the subgoals, agents playing the six other roles depicted in Figure 4 are required. First, an agent playing the Internal Components Assembler role puts together internal components of a cell phone. After that it passes the intermediate product to an Internal Components Tester. If the intermediate product does not pass the tests, an Engineer will fix it and return the product to an Internal Components Tester for an additional iteration of the same tests. After the tests have been passed, an Internal Components Tester sends the intermediate product to a Cover Assembler who equips the cell phone with display and covers. Thereafter it passes the final product to a Cell Phone Tester for ultimate testing. As previously, if a cell phone does not pass the tests, an Engineer will identify and fix the problem. If a cell phone has passed all the tests, it will be forwarded to a Visual Checker for final visual checking. Please note that goal models, such as the one represented in Figure 4, do not prescribe any temporary sequence of achieving subgoals.

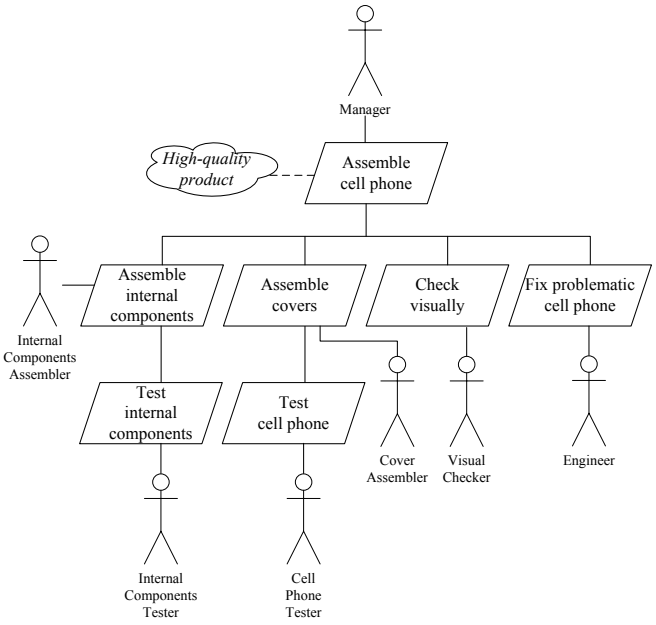

Figure 4. Goal model of assembling cell phones

Let us now suppose that the cell phone industry has new requirements for new cell phones with cameras. A sociotechnical multi-agent system has to be as flexible as possible to adjust to new processes and incorporate new goals and roles if needed. The production process of cell phones with cameras needs more attention as compared to the production process of "ordinary" cell phones. In the new requirements described as a goal model in Figure 5, this is reflected by the new goal of checking a camera and the new Camera Checker role attached to it. As a result of this change in the requirements, an agent playing the Camera Checker role joins the multi-agent system. After this, the multi-agent system continues to function as the production line described in the previous paragraph. The only difference is that an agent playing the Camera Checker role checks cameras before the ultimate visual checking activity is performed. If this new agent can inform the other agents of the system about its capabilities, the interactions, behaviours, and knowledge of just the *affected agents* in the supply chain will change accordingly. There is no need for all the agents of the system to be aware of a new agent playing the Visual Checker role. The whole system functions perfectly well when the new agent knows what to do and who to interact with and only some agents are aware of the new agent. In a similar way, all agents do not have to be aware of the overall

goal of the system. For example, a Visual Checker does not have to know that the purpose of the system is to assemble cell phones. The system operates very well when a Visual Checker only knows how to control cameras and who to interact with.

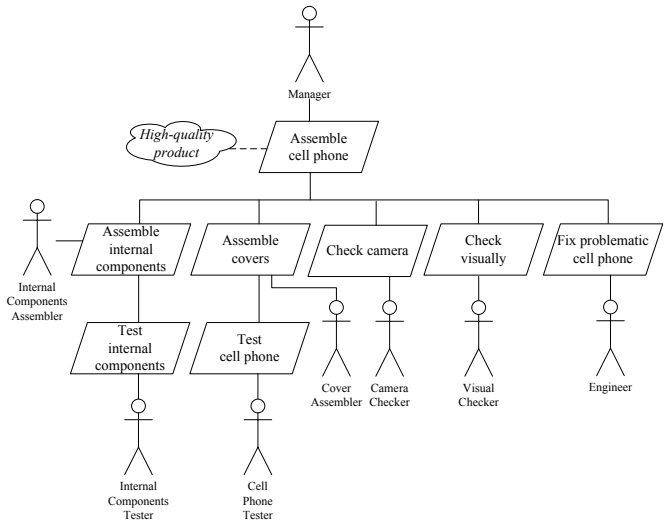

Figure 5. Goal model of assembling cell phones with cameras

Bottom-up iterative design organized in the way described in this section results in *evolutionary* multi-agent systems. In such systems, all the agents do not have to know what each of them knows. A multi-agent system functions properly when each agent is as a minimum aware of its own objectives and about the means of achieving them.

5. RELATED WORK

Multi-agent systems may have complex hierarchies of agents with various capabilities that affect the interactions between the agents. Horling & Lesser (2005) describe the following organisation types of multi-agent systems: Hierarchical Organization, Congregation, Agent Society, Holorchical Organization, Coalition-based Organization, Team-based Organization, Agent Federation, Market, Matrix Organization, and Compound Organization. These organization types reflect that multi-agent systems can use different schemas for interactions. We plan to apply in our work the organisation types of multi-agent systems proposed by Horling & Lesser (2005).

Sims, Corkill, & Lesser (2008) propose a bottom-up organization design as a process that accepts organizational goals, environmental expectations, performance requirements, role characterizations, and agent descriptions and assigns roles to each agent. The difference from our work is that while their approach is about engineering *closed* systems in the sense that all the goals and roles of the system are completely specified beforehand, our approach is for engineering *open* systems where new goals and roles may be added at any time.

6. CONCLUSIONS AND FUTURE WORK

In contemporary complex sociotechnical systems it is not feasible to possess all the information about the environment and to keep this information continuously updated. To reflect this, we proposed in this article an iterative bottom-up development approach of sociotechnical multi-agent systems. Such iterative development is flexible and adaptive. Therefore it is easy to adapt to a rapidly changing environment. In the near future, we plan to complement goal models for representing requirements by role and domain models and try our approach out in series of experiments.

The problems that have to be solved in the future are working out a formal language for describing the goals, roles, and domain entities. The changes in requirements could then be represented in that language. Also, we need to devise a formal language for describing capabilities of agents in the system and work out specific protocols how an agent entering the system can negotiate with other agents about coordination.

ACKNOWLEDGEMENTS

The authors thank for the support the research project "Model-based Creation and Management of Evolutionary Information Systems" (SF0140013s10) funded by the Estonian Ministry of Education and Research.

REFERENCES

Horling, B. & Lesser, V. (2005). A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review,19*, 281-316.

Luhmann, N. (1995). *Social Systems*. Stanford University Press, California.

Popper, K. (1964). *The Poverty of Historicism*. Harper & Row, New York.

Prigogine, I. (1997). *The End of Certainty. Time, Chaos and the New Laws of Nature*. The Free Press, New York

Sims, M., Corkill, D. and Lesser, V. Automated Organization Design for Multi-agent Systems. *Autonomous Agents and Multi-Agent Systems, 16*(2). 151-185.

Sterling, L. & Taveter, K. (2009). *The Art of Agent-Oriented Modeling*. Cambridge, MA, and London, England: MIT Press.