



University of
Calgary

Agent-based Software Engineering

Seng 609.22 Fall 2003

Instructed by Dr. Behrouz Homayoun Far

Agent-based software engineering

Tutorial Report

Reported by Jian Huang

Agent-based Software Tutorial Report

Jian Huang

Department of Electrical & Computer Engineering

University of Calgary

2500 University Dr. N.W.

Calgary, Alberta T2N 1N4, Canada

huangj@enel.ucalgary.ca

Abstract

Agent-based approach in software engineering(SE) is regarded as a promising technique in development software. This method could be adopted in many context, such as expert systems, AI, web services architecture, and so on. This report wants to discuss why agent-based approach becomes mainstream in SE by revealing inherent properties of agent. The drawbacks of this approach are introduced as well.

Keywords

Agent, Agent-based SE, Agent interactions

1.Introduction

Before introducing agent-based software engineering term agent should be expressed first. There have lots of definition on agent. In this paper the following definition on agent is adapted. By an *agent*, we mean a system that enjoys the following properties[1, pp116-118]:

- *autonomy*: agents encapsulate some state(that is not accessible to other agents), and make decisions about what to do based on this state, without the direct intervention of humans or others;
- *reactivity*: agents are situated in an environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the INTERNET, or perhaps many of these combined), are able to perceive this environment(through the use of potentially imperfect sensors), and are able to respond in a timely fashion to changes that occur in it;
- *pro-activeness*: agents do not simply act in response to their environment, they are able to exhibit goal-directed behavior by taking the initiative.
- *Social ability*: agents interact with other agents(and possibly humans) via some kind of agent-communication language, and typically have the ability to engage in

social activities(such as cooperative problem solving or negotiation) in order to achieve their goals.

In order to clearly understand the properties of agent let's think about some examples of agents.

- The animated paperclip agent in Microsoft Office
- Computer viruses(destructive agents)
- Trading and negotiation agents(e.g. The auction agent at ebay)
- Thermometer controller

All of above examples illustrate properties of agent.

Being a new research field, agent-based software engineering has become as a mainstream software engineering paradigm. This tutorial report wants to explain the reason why agent-based software engineering is regard as a mainstream in software engineering.

The remainder of the report is structured as follows. Section 2 discuss relationship between agent and other disciplines. Section 3 explains why agent-based techniques are likely to succeed and make it into the mainstream of software engineering. The drawback of adopting an agent-based approach will be discussed in section 4. Finally, a conclusion will be given in section 5.

2. Relationship between Agent and Other Terms

2.1 Agents and AI

The discipline of intelligent agents has emerged largely from research in AI. In fact, an intelligent agent could be considered AI product. But it is important to distinguish between the broad intelligence that is the ultimate goal of the AI community, and the intelligence we seek in agents[2]. The intelligence in agent is represented by autonomy property of agent. In some domains the learning capability of agent is also regarded as its intelligence. AI techniques are often the most appropriate way of building agents[2].

2.2 Agents and Expert Systems

An expert system is one that is capable of solving problems or giving advice in some knowledge-rich domain[3]. A classical example of an expert system is such a system which assist physicians in the treatment of human beings. The most distinction feature between expert systems and agents is the expert system does not communicate with any environment. For example, agent system could get information from sensors but expert system relies on human beings. Moreover, expert systems are not required to operate in real-time context. Finally, expert systems are not capable of co-operating with other agents. However, building expert systems can adopt agent approach.

2.3 Agents and Objects

Object-oriented development techniques are boom recently and are regarded as 'best practice' in industry and academic community. The most difference between agents and object is that in traditional object-oriented programs, there is a single thread of control[2]. On contrary, agents are executing entities currently. But object is a event-drive model. It reacts according to outside inputs. Object model is a static model of real-world while agent is a dynamic model of real-world. Agent is a rational decision making system: we require an agent to be capable of reactive and pro-active behavior, and of interleaving these types of behavior as the situation demands[2]. The object-oriented community has not addressed issues like cooperation, competition, negotiation, computational economies, and so on, that form the foundation for agent systems[4].

2.4 Agent-based SE and Agent-Oriented Development

Agent-Oriented Development is an extension of Object-Oriented Development. Development process covers the full process of development, including requirement specification, design, and programming. Agent-based SE also covers issues such as re-use and maintenance of the agent-system in addition to the development of the system itself [5].

3.Mainstream of SE

There are two main reasons for believing that agent-based techniques will become widely adopted. First, agent-based approach can be regard as a natural step in the evolution of a whole range of approaches to software engineering. Secondly, agent-based techniques are the ideal computational model for developing software for open, networked system, and loosely coupled environment.

Let's trace the development of programming models from assembly languages, procedural and structured programming, object-oriented programming, component-based, to design pattern, and software architectures. Firstly, the language conceptual basis has been changed from underlying machine architecture determining to problem domain determining. The agent-based world view is perhaps the most natural way of characterizing many types of problem. Agent is knowledgeable, autonomous, situated, and interactive software entity. Problem domain relative is agent inherent characteristic. Secondly, the basic building blocks of the programming models exhibit increasing degrees of localization and encapsulation[6]. Agent follows this trend by localizing purpose inside each agent, by giving each agent its own thread of control, and by encapsulating action selection. Thirdly, re-use mechanisms are being provided. In programming language, re-use mechanisms encompass code, library, component, design, and architecture. Agents enable whole subsystems and flexible interactions to be re-used rather than re-use of subsystem components and preordained interactions. Agent designs and implementations are re-used among applications. Consider, for example, the class of agent models that has beliefs(what the agent knows), desires(what the agent wants) and

intentions(what the agent is doing) at its core. Such Belief-Desire-Intention(BDI) architectures have been used in a wide variety of applications including air traffic control, process control, simulation, fault diagnosis, transportation, and scientific data interpretation. Flexible patterns of interaction such as the Contract Net Protocol have been re-used in applications. In addition to these two kind of re-use, agents are capable to re-use non-agent software. The technique is to wrap conventional software to agent. That means the wrapper presents an agent interface to the other agent and thus from the outside it looks like an agent. On the non-agent software inside, two-way translation is performed. In summary, agent-based techniques represent a natural progression of current software engineering thinking and, for this reason, the main concepts and tenets of the approach should be readily acceptable to mainstream practitioners[7].

For open, networked system, N-M model[8], the dominant software model needs to be based on synthesis or construction, rather than decomposition or reduction. The whole software system is combined with independently developed software entities which are communicating with one another. Standing from such perspective, the software computational model should include the following properties:active and autonomous, reactive and proactive, knowledgeable and learning. Of course, the desired computational model is suitable for agent-based. For example, web-services model needs to be agent-based.

4. Drawbacks of Agent-based SE

Two major drawbacks associated with agent-based approach are shown as below.

- The patterns and the outcomes of the interactions are inherently unpredictable.
- Predicting the behavior of the overall system based on its constituent components is extremely difficult because of the strong possibility of emergent behavior[7].

Although the flexibility of agent interactions has many advantages when engineering complicate systems, the disadvantage is that it leads to unpredictability in the run-time system. On one side, decisions about the number, pattern and timing of interactions depend on a complex interplay of the agents' internal state, the agent's perception of the environment and organizational context that exists when the decision is made. It is difficult for agent to make predictions about the system's interactions. On the other side, there is a de-coupling, and potentially a considerable degree of variability, between what one agent first requests through an interaction and the outcome that eventually ensues. The request may be immediately accepted, refused completely, or modified through some form of social interchange. In summary, in the general case, both the nature and the outcome of an interaction cannot be determined at the onset.

It has long been recognized that interactive composition-collections of processes(agents) acting side-by-side and interacting in whatever way they have been designed to interact-results in behavioral phenomena that cannot be deconstructed solely in terms of the behavior of the individual components. Such emergent behavior is a consequence of the

interaction between components and given their sophistication and flexibility in agent systems, it is clear that the scope for unexpected collective behavior is considerable. In certain situations(e.g., social simulations and market systems) emergence is not necessarily a bad thing since the ensuing behavior is a more accurate model of the problem being addressed.

4. Conclusion

Agent-based system is a promising development. It will be successfully applied in expert systems, web services system, AI field, and so on. This tutorial report just gives a high level concept on why agent-based software engineering becomes mainstream in SE and drawbacks in adopting this approach in SE. Lots of other fundamental problems of agent-based SE have not been discussed, such as specification, implementation, and verification from the point of view of agent-based systems. I hope it will stimulate your interest on agent-based systems.

References

- [1] M.Wooldridge and N.R. Jennings. Intelligent agents: Theory and practice. The Knowledge Engineering Review, 10(2):115-152, 1995
- [2] Michael Wooldridge, “Agent-Based Software Engineering”, September 19, 1997
- [3] P.Jackson. “Introduction to Expert Systems”, Addison-Wesley: Reading, MA, 1986
- [4] A.H. Bond and L. Gasser, editors. “Readings in Distributed Artificial Intelligence”, Morgan Kaufmann Publishers, San Mateo, CA, 1988
- [5] Amund Tveit, “A survey of Agent-Oriented Software Engineering”,
- [6] H.V.D. Parunak, Industrial and practical applications of DAI, in: G. Weiss(ED.), Multi-Agent Systems, MIT press, Cambridge, MA, 1000, pp.377-421
- [7] Nicholas R. Jennings, “On agent-based software engineering”,
- [8] Y.Wang(2000/./03), Lecture Notes on Theoretical Foundations of Software Engineering