

 UNIVERSITY OF CALGARY	Course Number: SENG 609.22	Course Name: Agent-based Software Engineering
	Session: Fall, 2003	Department: Electrical and Computer Engineering
		Document Type: Tutorial Report

Agent-Based Software Analysis and Design

Edward Dantsiguer
SENG 609.22
Fall 2003

Tutorial Paper 1

Abstract

With the emergence of agent-based software paradigm, the need for tools and techniques to perform analysis and design of such systems became apparent. This paper takes a brief look at a small sample of agent-oriented analysis and design techniques. This paper is not meant as a comprehensive overview of available approaches.

Introduction

Software design is an issue of fundamental importance regardless of what approaches are used. Solid design is essential regardless of whether the techniques that implement this design, and the design itself, are based on functional or object-oriented programming or agent-based approaches. When developing agent-based systems, the central role of good system design does not diminish. Careful planning and documentation never go out of style regardless of what the approach, methodology and technology of the day happens to be (unless it is Agile, of course). For more information on agents, please refer to Jennings et al, 1998 [1].

Background

The software development lifecycle (in its simplified form) consists of the following sequential steps: requirements gathering, analysis, design, implementation, and testing. In modern variants of this process, the steps are iterated to improve software predictability, and quality. This paper makes no efforts to address requirements gathering, implementation (coding) or testing phases of the software development lifecycle.

With the emergence of agent-oriented technologies and the promise they bring to improving the way software is built, there have been numerous agent analysis and design approaches developed. Advocates of agents claim many benefits from their use. These include better ways to structure applications, improvement in programming language abstractions for complex inter-agent communication, and a natural metaphor for human-computer interaction (Jennings et al, 1998) [1].

There are two schools of thoughts when it comes to the issue of designing agent-based systems. One stipulates that agent-oriented software should be analyzed, designed and developed independently from object-oriented techniques and methodologies. Tropos would an example of such approach (please see Mylopoulos, 2001 [3] for more details). Others take an approach of extending existing OO techniques to support agent-based concepts (Henderson-Seller, B, 2003) [2]. This means mixing agent-based ideas with object-oriented approaches.

Most, but not all, of today's agent-based approaches are centered around the analysis phase of the software development lifecycle. This means that design

activities have to be performed using other approaches, such object-oriented techniques.

There are two good reasons for mixing agent-based analysis techniques with object-oriented design methods. First of all, agents are at a higher level of abstraction than, say, for example, objects, or any other software implementation technique or approach. This means there has to be a transition to a more detailed, lower level of abstract phase between analysis and implementation. This then means that agents require a lower-level of abstraction counterpart, come design time.

The second reason is lack of agent-oriented design tools that can aid in this process. Although some such tools exist, they are not commercial strength and cannot be applied widely.

OPEN Process Framework

Object-Oriented Process, Environment and Notation Process Framework (OFP) (see Graham et al, 1997 [4]) is a suitable, existing infrastructure upon which a support facility for multi-agent design could be built. It is less rigid and restrictive than conventional object-oriented approaches. By using OFP, process components are selected from an existing repository and the required methodology is constructed using identified construction and tailoring guidelines. OFP offers a useful starting point as it provides the building blocks that are required to tailor a new methodology. This environment is conducive to adding the needed support for design of intelligent agents.

Numerous extensions have been made to OFP to improve its suitability and usefulness to agent-based software development. Please see Debenham, J. and Henderson-Sellers, B., 2003 [6] as well as Henderson-Sellers, B., Giorgini, P., and Bresciani, P., 2003 [7] for more details.

Criticism of OFP

Although OFP builds on OO principles to provide a framework for agent-based system development, it lacks a number of important aspects that are central to good agent-based systems. Agents, unlike objects, have roles and responsibilities. The main problem with OFP is that even though it makes an attempt to represent roles, it lacks the more sophisticated notions and aspects of roles in agent technology.

GAIA Approach

GAIA (Wooldridge, M., Jennings, N.R. and Kinny, D. (2000) [5]) methodology is another way of looking at agent-based systems. It consists of two main parts: analysis and design. GAIA offers an extensive analysis approach which includes identifying the required agent types, their roles, responsibilities, permissions, and interactions.

The system is viewed as a society of agents. Each agent has a role to play in that society. These roles are defined by three attributes: responsibilities, permissions and protocols. Responsibilities will dictate functionality, as the agent is responsible for accomplishing the tasks defined in its “responsibilities” attribute. Permissions are coupled with responsibilities. They are the “tools” or “keys” required to accomplish required responsibilities. Protocols define the way that a given role can interact with other roles in the community. Protocols form an area of agent interaction.

The second part of GAIA is design. Design is subdivided into three models: agent model, service model and acquaintance model. Agent model documents different agent types that the system will use and also lists the agent instances that will be instantiated from these types. Service model documents major services that each agent type will provide. Lastly, acquaintance model defines the communication links that exist between different agent types.

Once analysis and design phases are completed, GAIA relies on other techniques to complete the design (transform it into the required level of detail sufficient for implementation purposes) and implement the system. Object-oriented techniques would be one such choice.

Criticism of GAIA

It appears that the design process, as defined by GAIA, is not really a design process in the conventional meaning of that term. It is more of an extension of analysis or a form of pre-design. It does not appear to add a lot of value to the overall software development process. Agent model is not too different from Roles model. Services model is not too different from Responsibilities model. Although a slight step towards a more detailed representation is taken in each case, the step is too small to justify the required effort.

Recommendations

Perhaps a more realistic and cost/time effective approach to developing agent-based software systems would be taking the analysis process provided by the GAIA approach directly followed by design techniques provided by object-oriented design. This approach can capitalize on the benefits provided by GAIA analysis phase in identifying the needed agent types, their roles, responsibilities, permissions and interactions while using an established OO-methods to design the system from that point forward.

Conclusion

There is no “one fits all” approach to designing agent-based systems. There is no universally applicable approach that works for every project and every situation. One has to make decisions based on the situation, project, available funds, project timeline and available technology.

When it comes to agent-based software development, it is important to realize that this technology is still in its infancy. It would be more convenient to use agent-based approaches for both analysis and design phases of software lifecycle. However, agent-based design techniques are not sufficiently developed to provide quality help that would be required to properly design fully functional, quality software in a commercial environment. Before these techniques mature, it would be more practical to rely on established designed methods such as object-oriented approach to aid in the design process.

When agent-based analysis techniques are coupled with object-oriented design techniques, the combination makes for a usable approach to realistic software development. This allows more companies to capitalize on the advantages the agent paradigm provides. This brings more attention to agent-based methodologies and promotes further work in the area of agent-based design. With time, affordable, user-friendly and powerful tools and techniques for agent-based design will become available.

References

1. Jennings, N. R., Sycara, K. and Wooldridge, M (1998). A Roadmap of Agent Research and Development. *Int. Journal of Autonomous Agents and Multi-Agent Systems*, 1 (1), 7-38.
2. Henderson-Seller, B., Gorton, I. (2003). *Agent-based Software Development Methodologies*.
3. Mylopoulos, J., Kolp, J. and Castro, J. (2001). UML for agent-oriented software development: the Tropos proposal, in «UML»2001 – The Unified Modeling Language (eds. M. Gogolla and C. Kobryn), LNCS Vol 2185, Springer-Verlag, 422-441.
4. Graham, I., Henderson-Sellers, B. & Younessi, H. (1997). *The OPEN Process Specification*. Harlow, UK: Addison-Wesley, 314pp.
5. Wooldridge, M., Jennings, N.R. and Kinny, D. (2000). The Gaia methodology for agent-oriented analysis and design. *J. Autonomous Agents and Multi-Agent Systems*, 3, 285-312.
6. Debenham, J. and Henderson-Sellers, B., *Designing agent-based process systems | extending the OPEN Process Framework*, Chapter VIII in *Intelligent Agent Software Engineering* (ed. V. Plekhanova), Idea Group Publishing (2003) 160-190
7. Henderson-Sellers, B., Giorgini, P., and Bresciani, P., *Enhancing Agent OPEN with concepts used in the Tropos methodology* (2003)