 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

Examining the Multiagent Systems Engineering Methodology

SENG 609.22
Kendra Hamilton
December 2003

Abstract

As agent systems become more prevalent, methodologies are needed to help analysts and designers in the creation of new systems. Wood and DeLoach (2000) propose the Multiagent Systems Engineering Methodology (MaSE) as a practical, formal methodology for the creation of multiagent systems. This work provides an overview of MaSE and provides a critic of the methodology.

Introduction

According to its designers, Wood and DeLoach (2000), Multiagent Systems Engineering Methodology (MaSE) was developed as a methodology that covered the complete software lifecycle of multiagent systems. Their choice to create this methodology was due to their perception that contemporary agent system methodologies were limited to covering only the high-level analysis portion of the life cycle, leaving a gap between analysis and design.

This paper begins by describing the steps covered in MaSE. It proceeds by reviewing the MaSE methodology in terms of how well it considers and attempts to deal with specific agent qualities such as roles, autonomy, and reactivity.

Description of MaSE

As seen in Figure 1, MaSE is a top-down methodology made up of seven steps, which covers the analysis and design portions of the software lifecycle. During the seven steps, a total of nine models are produced. The MaSE methodology begins with system requirements as input and ends with deployment as the last step. Each step will be further discussed in detail.

As an agent system methodology, MaSE does not specify a particular system architecture, agent architecture, programming language, or communication protocol to be used. These choices are left open to the system designer.

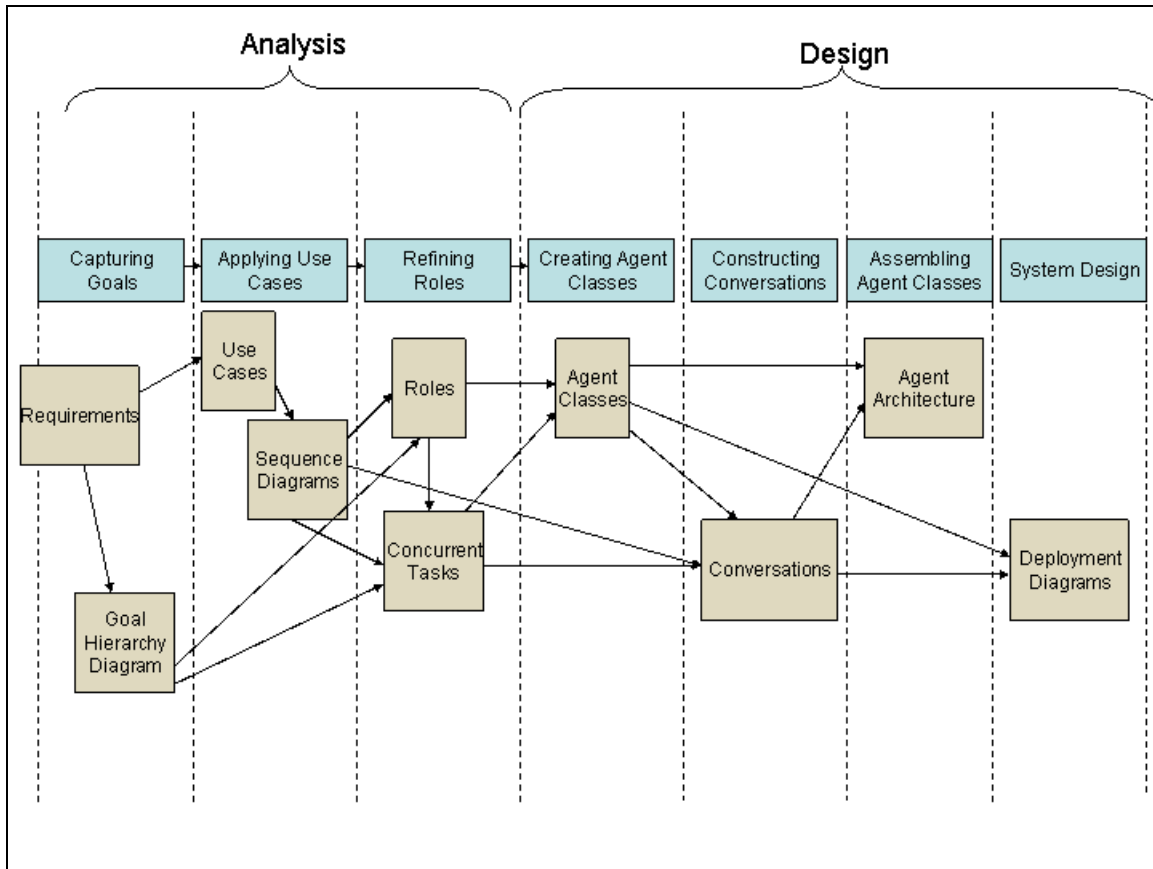


Figure 1 MaSE Methodology (DeLoach et al., 2000)

1. Capturing Goals

As input, the Capturing Goals step assumes that the system requirements specification has already been completed and can act as input.

This step is concerned with transforming the system specification into goals. It is important to note that the goals being defined are not project level goals, such as “save cost”, or agent level goals, rather they are high level system objectives or processes which the system must support. Wood and DeLoach (2000) do not specify how to identify and subdivide the system goals from the system requirements specifications, but they do indicate that the goals should be documented in a Goal Hierarchy Diagram. The Goal Hierarchy Diagram is a visual means of presenting goals in a hierarchical format where all goals at the same horizontal level should be similar in scope and where sub-goals can trace their functionality up the hierarchy.

2. Applying Use Cases

The second step of the MaSE methodology is made up of two parts. The first part is the creation of use cases from the original system specifications where use cases are those found in traditional object-oriented methods. Secondly, sequence diagrams are created

from the use cases produced; there should be at least one sequence diagram created from each use case. The sequence diagram, also traditionally found in object oriented methods, models the messages passed between object classes or agent roles. If the agent role which sends the message belongs to a different age and class then the agent role which receives the message, the communication is considered a conversation.

3. Refining Roles

The third step of MaSE is to transform goals into roles. Wood and DeLoach (2000) suggest that in most cases, each of the lowest level goals in the Goal Hierarchy Diagram should become a role, though they acknowledge that in cases where there is a strong similarity between two goals, the system designed may want to combine the goals into a single role. Furthermore, Wood and DeLoach (2000) specify that each interface with an external source or with a mobile agent requires a role to be associated with it. Roles are documented in a Role Model with a few modifications:

- Goals are listed with roles
- Tasks are associated with one and only one role and are shown as ovals with the task name as a label. The association is depicted as a line between the task and the role.
- Arrowed lines between tasks suggests a possible communication path.

Once the Role Model is constructed, tasks are modeled as state diagrams.

4. Creating Agent Classes

Once roles have been defined, the next step is to create agent classes, which are a combination of roles and conversations. Wood and DeLoach (2000) suggest that normally there should be a one-to-one correspondence between agent roles and agent classes though the designer may want to combine several roles into one class or to split a role into several classes. It is suggested that if two agent roles conduct a lot of conversations, if the conversations occur frequently, or a lot of data is transmitted during the conversation, the designer should consider merging the roles into a single agent class.

The final work product of the Create Agent Classes step is the Agent Class Diagram. In the Agent Class Diagram, agent classes are depicted as boxes with agent class names inscribed in the box. Conversations between agent classes are depicted with arrows between the boxes and the conversation names written close to the arrows.

5. Constructing Conversations

The fifth step of MaSE is defining the coordination protocol between two agents, the initiator of the conversation and the responder. Two Communication Class Diagrams are

created for each conversation, one for the initiator and one for the responder. In the diagram, each side of the conversation is shown as a finite state machine.

6. Assembling Agents

Assembling agents involves defining the internal architecture of the agent classes previously defined. Five architectural templates are suggested: Belief-Desire-Intention, reactive, planning, knowledge based, and user defined architecture. Each of the architectural templates suggested has a set of components associated with it. The designer may choose to use pre-existing components or define their own. Each component may be further subdivided into other components. Components are joined by inner-agent connectors when the calling and the called components reside in the same agent. Components are joined by outer-agent connectors if the calling and the called components reside in separate agents.

7. System Deployment

The final step of MaSE involves the instantiation of agent classes. A Deployment Diagram is created to represent the number of instances of agents, types of agents, and locations of the agents in the system. Similar to the Agent Class Diagram, the Deployment Diagram incorporates the instances of agents, which are modeled as three-dimensional boxes, and conversations, which are modeled as lines connecting the boxes. If there are multiple instances of the same class, the instances are encapsulated by a rectangle with a dashed line border.

Review of MaSE

It is interesting to note that DeLoach (DeLoach, 1999) considers MaSE a “further abstraction of the object-oriented paradigm where agents are at an even higher level of abstraction than typical objects....[Agents] may or may not possess intelligence.” This approach to agent systems can be seen in MaSE. Not only are most work products based from object oriented methods, but little (if any) emphasis is placed on some of the more defining characteristics of agents such as autonomy, pro-activity and the ability to learn. As a result, the scope of agent systems, which MaSE would be appropriate for, is rather limited. Moreover, this methodology can blind analysts and designers to the potential of agent systems, resulting in systems that are basically object oriented where the “object” is a role.

The MaSE methodology is a well defined step-by-step approach to system analysis and design and results in well documented software life cycle phases. However, like all other methodologies, the ability and experience of the system analyst and designer have a significant impact on the success of the methodology. This is due to the lack of suggestions or help given by MaSE on how to construct the models. The models only

provide a visual language to document the analysis and design and do not suggest how to do the actual analysis or design. For example, the analyst must determine how detailed the goal hierarchy should be, which goals belong to which roles, how many roles there should be, how many agent classes there should be...

As opposed to what normally might be expected, use cases are created in MaSE after the system requirements have been created and after the requirements have been broken down into a task hierarchy. This may lead to some difficulty for the analyst, as use cases are generally at a much higher level of abstraction and would normally precede the creation of goal hierarchy. As a result, use cases could be made to fit into the goal hierarchy. This approach would have a tendency to produce more of a procedural program than a system made up of autonomous agents that react to the environment around them and make decisions in order to achieve their goals.

In order for agents to communicate successfully, a common ontology must be shared between them. DiLeo et al (2002) have addressed this issue and proposed the addition of an eighth step "Building Ontology" between the Applying Use Case step and Refining Roles Step, reasoning that the ontology, a necessary part of agent communication, can be defined using the Goal Hierarchy Diagram, Use Cases, and the Sequence Diagram and is needed to create tasks in the Refining Roles step since tasks often involve parameter passing.

Conclusion

MaSE provides a step-by-step approach to multiagent system and design where work products from one step flow into the work products in successive steps. However, characteristics such as autonomy, pro-activity, and learning are virtually ignored by the methodology. As a result, while the objective of MaSE was to create a complete agent system development methodology, it is rather limited in its definition of what an agent is and therefore is of limited benefit to actual multiagent system design. Further work and modifications to MaSE need to be done in order for it to be successful in designing agent systems.

Bibliography

DeLoach, Scott, et al., "Exploiting Agent Oriented Software Engineering in Cooperative Robotics Search and Rescue", *International Journal of Pattern Recognition and Artificial Intelligence*, Vol 17, No 5, 2003.

DeLoach, Scott, "Multiagent System Engineering: A Methodology And Language for Designing Agent Systems", *Agent-Oriented Information Systems*, Seattle, Washington, May 1999.

DiLeo, Jonathan, et al., "Integrating Ontologies into Multiagent Systems Engineering", *Proceedings of the Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems*, AOIS 2002.

Wood, Mark; DeLoach, Scott, "An Overview of the Multiagent Systems Engineering Methodology", *Proceedings of the First International Workshop on Agent-Oriented Software Engineering*, Limerick, Ireland, June 2000.