Multiagent Systems Engineering (MaSE) – An Introduction

written by Arnon Sturm

The Multiagent Systems Engineering (MaSE) is a general purpose methodology for developing multi-agent systems that is founded on the basis software engineering principles. MaSE divides the development process into two major phases: the analysis phase and the design phase. For each phase MaSE provides a set of stages need to be performed. Figure 1 presents the development process proposed by MaSE. The analysis phase consists of the following stages: capturing goals, applying use cases, and refining roles, and the design phase consists of the following stages: creating agent classes, constructing conversations, assembling agent classes, and system design. In the following we elaborate on the various stages.

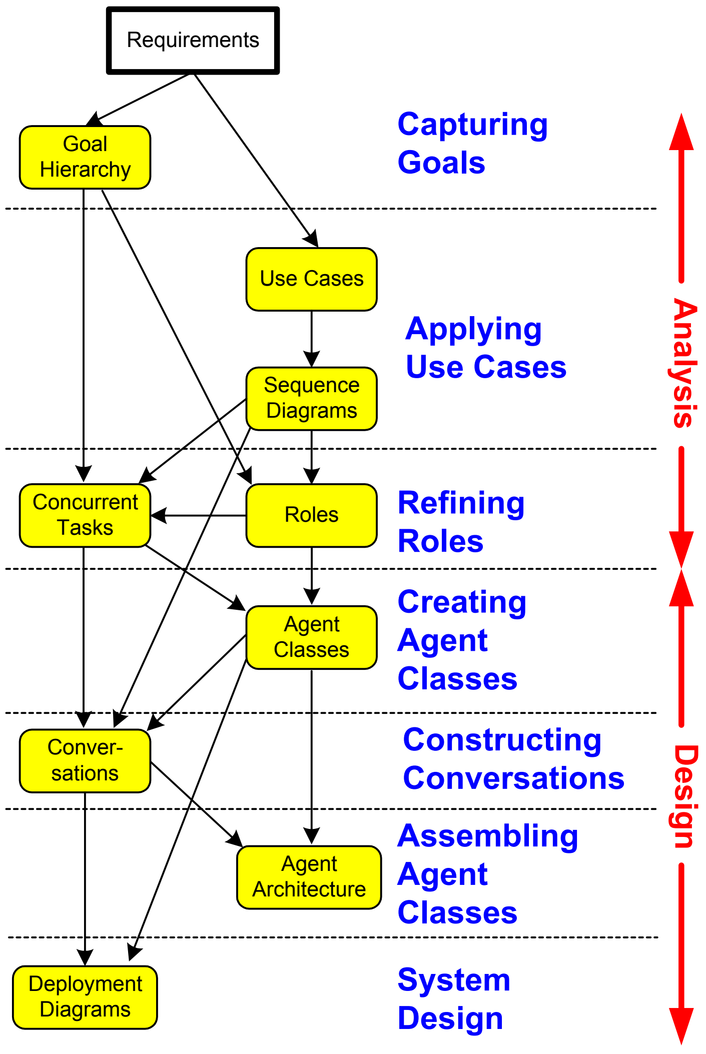

Figure 1. MaSE development stages

The Analysis Phase

The purpose of the analysis phase is to provide a set of roles whose tasks meet the system requirements, i.e., specifying *what* the system should do. According to MaSE, the analysis phase consists of the following stages: capturing goals, applying use cases, and refining roles.

- Capturing Goals

In this stage the system goals are being elaborated specified from the system point of view and not from the user point of view. In MaSE, a goal is an abstraction of a set of functional requirements. The stage of capturing goals comprises two sub-stages: identifying the goals and structuring them in a hierarchy, in terms of goal and their sub-goals is being constructed.

- Applying Use Cases
  In this stage the system use cases are being specified. It is divided into two sub stages: the creation of use cases and the creation of the sequence diagrams. A use case is a set of interactions. It describes the general system behavior, i.e., what the system should do. The transformation from the use cases specification to sequence diagrams is straightforward; each entity becomes a role and information passing becomes an event (or a message).
- Refining Roles
  In this stage the system functional decomposition is determined. It is done by producing a set of roles and their associated tasks. This stage consists of two sub-stages: building the role diagram and specifying the tasks' behavior. The sources for that stage are the goals determined in the first stage and the sequence diagrams created in the second stage. A role can be derived from the roles determined during the sequence diagram creation or can be formed directly from the goals hierarchy. At any rate, each role should be associated with at least one goal indicating that the role is responsible to achieve it (or them). In addition, each role should have tasks that realize its goals. When building the role model the interactions between the roles are specified by connecting their tasks. These links, which are derived form the sequence diagrams created in the previous stage, depict the protocols among the role. Upon completion of the role model each task should be specified. The semantics of the tasks within MaSE is that each task runs concurrently and independently of the other tasks within the model. The task model in MaSE are specified using a finite state automaton and is called concurrent task diagram.

The Design Phase

The purpose of the design phase is to specify the way the system-to-be should behave and be constructed. That means, specifying *how* the system will achieve its goals. The design phase consists of the following stages: creating agent classes, constructing conversations, assembling agent classes, and system design.

- Creating Agent Classes
  In this stage, the overall multi-agent system architecture in terms of agent and the conversations among them is determined. Agent classes are created from the roles define in analysis phase by assigning roles to agents. Each agent is associated with at least one role. In addition to the agent classes, the conversations among them are also specified utilizing the protocols defined in the analysis phase (the links among tasks within the role model).
- Constructing conversations
  In this stage, the designer defines the coordination protocols (i.e., conversations) between agent couples. In particular, two communication class diagrams are defined for each conversation. One diagram specifies the initiator behavior during that conversation and the second one specifies the responder behavior during that conversation. The communication class diagram is designed using a finite state automaton. The detailed design of each conversion is derived from the relevant concurrent task diagrams.

- Assembling Agent
  In this stage the internal architecture of the agents is being specified. One can use its own architecture to build an agent (e.g., BDI) or convert the tasks from the previous stage into components. The agent architecture consists of the components and the relationships among them. These components can be specified recursively, i.e., a component may have sub-components, and may have a finite state automaton which defines its behavior.
- System Design
  This stage is aim at depicting the physical system architecture and the distribution of the various agent classes' instances within that architecture.

Development Process

- Implementation Issues
  MaSE also provides a class library called agentMOM that can be utilized for code generation. Each of the MaSE design artifacts may be a specialization of the relevant class from the agentMOM library.
- Development Process
  MaSE follows the iterative development process principles. It means that any stage can be repeated many times till the final design is achieved and at any stage a designer may go back to a pervious stage.
- CASE Tool
  MaSE is supported by a CASE tool called agent Tool. This tool has the capability of performing the analysis and design activities determined by the MaSE process, perform the validation of the various models, generate automatic transformation of models, and generate skeleton code.
- Extensions
  MaSE also supports the specification of mobility aspects and the definition of ontology.