Chapter 2.6
The Agent-Oriented Methodology MAS-CommonKADS

Carlos A. Iglesias
Technical University of Madrid, Spain

Mercedes Garijo
Technical University of Madrid, Spain

ABSTRACT

This chapter introduces the main concepts of the methodology MAS-CommonKADS that extends object-oriented and knowledge engineering techniques for the conceptualisation of multi-agent systems. MAS-CommonKADS defines a set of models (Agent Model, Task Model, Expertise Model, Coordination Model, Communication Model, Organisation Model, and Design Model) that together provide a model of the problem to be solved. Each of the components of the model is a generic component for the sake of reusability. Readers familiar with object-oriented analysis will find it easy to apply most of the techniques of MAS-CommonKADS in the development of multi-agent systems and will be introduced to the application of knowledge engineering techniques for specifying the knowledge of the agents.

INTRODUCTION

MAS-CommonKADS is an agent-oriented software engineering methodology that guides the process of analysing and designing multi-agent systems. MAS-CommonKADS distinguishes several development phases: *conceptualisation*, where the system is conceived as a multi-agent system and where agent properties of the system are identified; *analysis*, where different models are developed in order to analyse the system from different points of view; *design*, where the different models are operationally focussed; and *development* and *testing*, which are not addressed explicitly in the methodology.

MAS-CommonKADS (Iglesias, 1998; Iglesias, Garijo, González, & Velasco, 1998) can be used in combination with other methodologies. For example, some of its conceptualisation tech-

niques, such as *Class-Responsibility-Collaboration* (CRC) cards (Beck & Cunningham, 1989; Wirfs-Brock, Wilkerson, & Wiener; 1990) and *User-Environment-Responsibility* (UER) techniques (Iglesias & Garijo, 1999) can be used for conceiving a system from an agent point-of-view and be combined with other methodologies such as *Rational Unified Process* (RUP) (Kruchten, 2000) or *eXtreme Programming* (XP) (Beck, 1999); or use another agent-oriented methodology. In the same way, every analysis model can be used in combination with another methodology.

MAS-CommonKADS has as one of its goals to be usable by professionals who want to include in their projects this new and exciting computation paradigm—agents. In this way, MAS-CommonKADS extends well-known modelling techniques, such as CRC cards, use cases, *Message Sequence Charts* (MSC) (ITU-Z.120, 1996) or *Specification and Description Language* (SDL) (ITU-T-Z.100, 1994) diagrams, with new perspectives driven by the agent metaphor.

This makes many of MAS-CommonKADS techniques easy to learn and practice. The recent addition of MSC and SDL diagrams to *Unified Modelling Language* (UML) (Salic, 2004) makes MAS-CommonKADS even easier to practice with standard object-oriented CASE tools that can be enhanced with stereotypes for some of the new modelling entities (for example, software actor or environment).

THE MAS-CommonKADS METHODOLOGY

The origins of MAS-CommonKADS come from CommonKADS (Schreiber et al., 1999), a well-known knowledge engineering methodology, and from object-oriented methodologies such as *Object Modelling Technique* (OMT) (Rumbaugh, Blaha, Premerlani, & Eddy, 1991), *Object-oriented Software Engineering* (OOSE) (Jacobson, Christerson, Jonsson, & Övergaard, 1992) and *Responsibility Driven Design* (RRD) (Wirfs-Brock, Wilkerson, & Wiener; 1990). In addition, it includes techniques from protocol engineering such as SDL and MSC. All these techniques are combined in order to provide support to agent developers.

MAS-CommonKADS is based on the models of CommonKADS extended and adapted to agent modelling, including the definition of a new model, the coordination model, for describing agent interactions.

The software development life cycle in MAS-CommonKADS follows the phases described below:

- **Conceptualisation:** Elicitation task in order to obtain a first description of the problem through the definition of a set of use cases that help to understand the system and how to test it.
- **Analysis:** The analysis phase determines the functional requirements of the system. It describes the system through the development of a set of models.
- **Design:** The design phase combines a top-down and bottom-up approach, reusing developed components and developing new ones, depending on the targeted agent platform. The design phase takes as an input the analysis models, which are then operationalised, that is, transformed into specifications (the design model) ready to be implemented. The internal architecture of every agent and the "network architecture" of the system are determined.
- **Development** and **testing:** Coding and testing tasks of the previously defined agents.
- **Operation:** Maintenance and operation of the system.

The methodology defines the following models (Figure 1):