Object-oriented Design and Implementation of the Multi-agent System AgentTeam*

Bora Ý Kumova

Dokuz Eylül University, Dept. of Comp. Eng., 35100 Ýzmir, Turkey
kumova@cs.deu.edu.tr; cs.deu.edu.tr/~kumova

Keywords. Object-oriented design; Distributed Systems; Multi-agent Systems

Background. In this work, we discuss the multi agent system, AgentTeam, from a software engineering perspective. AgentTeam is a framework for agent-based flexible management of distributed relational data; as stored in heterogeneous databases. In order to fit the inherently distributed application environment of the Internet, it is designed in an object-oriented fashion. It consists of domain-dependent system concepts and partially domain-independent software design concepts. In this contribution the domain-independent software design concepts are evaluated with respect to re-useability and scalability, and discussed in detail on the prototype, CourseMan. Re-useability means that a concept can be used in the design of systems in different application domains. Scalability means that a concept can be used multiple times inside a specific design, where concepts can be any design abstraction or implementation unit.

Software Design Concepts of the AgentTeam Framework. Class, Class Relationship, Task, Instance, Object, Object State, Module, Agent, Agent Knowledge Representation, Agent Communication Language (ACL), Agent Communication Structure, Agent Co-operation Structure, Agent Co-ordination Structure, Agent Behaviour, Agent Property.

We believe that these concepts are highly re-useable and are principally scalable, since they have been built independently from domain-specific concepts.

Class Structures of the CourseMan Prototype. SemanticNode, SemanticNet, Template, Topic, FIPA ACL, AgentCom, InferenceMechanism, Agent, UserAgent, CMUserAgent, TaskAgent, JavaServer, ResourceAgent, DatabaseAgent (*Fig.1*).

These are the basic classes implemented in CourseMan.

Object Structures of the CourseMan Prototype. The MAS CourseMan consists mainly of four agent types, which communicate with each other in a well-defined communication structure. Most of the relationships between objects have variable cardinality, where a concrete cardinality depends on the state of an agent. Each agent type can create one or more instances of FIPA ACL and AgentCom, in order to communicate

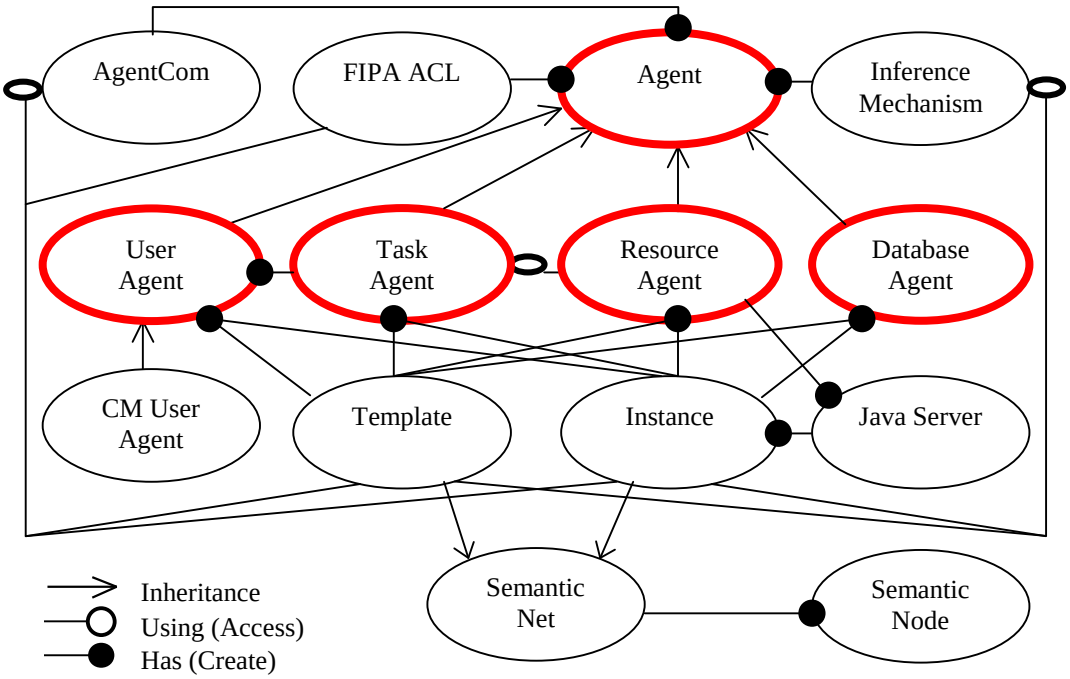

Fig.1: Class Diagram of CourseMan

with another agent. The role of a task agent is to determine for a given task, the necessary co-operation structures and to co-ordinate the communication.

State Transitions of a CourseMan Agent. In *sleeping*-state, the agent stays persistent, until requested to perform a task. If no task is scheduled, then a busy-*wait* state is entered. If data is requested from another agent, then the *communication*-state is entered. While performing a computation, the *evaluation*-state is entered.

Module Structure of CourseMan. The whole CourseMan system consists of two modules, one for client site, and one for server site. Scalability of modules is unrestricted for both, clients and servers. All code is written entirely in Java scripts. On the client side, the user agent is embedded in an HTML page as an applet.

All agents implement the AgentCom language interface to communicate with each other. Communication between agents, which is compiled to one code, is synchronous. Communication between agents over the network can be established optionally in asynchronous mode.

Some Scenarios. At run-time, the user agent dynamically creates one or more task agents, which establish the network connection to resource agents at server sites. To receive messages from database servers asynchronously, the user agent creates a server side skeleton at the client. This listens to the standard port of the RMI system.