Abstract. A competition always shows the performance of the participants. We have developed the JIAC IV agent framework over years now and took this as a chance to see where we stand. This paper describes our approach to the contest scenario from a software engineering point of view, i.e. how we would solve similar problems of complex and distributed nature.

1 Introduction

The JIAC IV agent team has been prepared by members of the Competence Center Agent Core Technologies of DAI-Labor at Technische Universität Berlin. We use the JIAC IV agent framework with accompanying toolkit, which have been created in the course of several projects at DAI Labor, intended for telecommunications and telematics services to be implemented quickly and effectively, and to be administered reliably.

2 System Analysis and Design

The Java Intelligent Agent Componentware agent framework (JIAC IV) comes with its own customised methodology and a number of tools integrated in the Eclipse IDE.

As shown in Figure 1, the development process starts with collecting domain vocabulary and requirements, which then are structured and prioritised. Second, we take the requirements with the highest priority and derive a MAS architecture by listing the agents and create a user interface prototype. The MAS architecture then is detailed by creating a role model, showing the design concerning functionalities and interactions. We then implement plans, services and protocols, which are plugged into agents during integration. Agents are deployed to (one or more) agent platforms and the application is ready to be evaluated. Depending on the evaluation we align and amend requirements and start the cycle again with eliminating bugs and enhancing and adding features until we reach the desired quality of the agent-based application.

The JIAC methodology is based on the JIAC meta-model. JIAC has explicit notions of goal, rule, plan, service and protocol. Knowledge written in JADL and AgentBeans written in Java constitute agent roles, which are plugged into
standard JIAC agents. The standard JIAC agent is already capable of finding other JIAC agents and their services, using infrastructure services and provides a number of security and management features.

For any part of the JIAC meta-model we provide an editor (source code as well as visual editor) in the JIAC IDE for easy agent and application development. Reuse is supported by a plugin that allows search and retrieval of components and solutions. A context sensitive help and a number of interactive tutorials complete the JIAC IV toolbox.

3 Single Agent Behaviour

We started collecting the simulation domain vocabulary and created the ontology containing such concepts as nuggets, gold-digger, grid cells, and so on. Listing 1.1 shows the GoldDigger category with its attributes. Furthermore, basic features such as the ability to communicate with the simulation server and a simple path-finding algorithm have been created.

```
(cat GoldDigger (ext TemporalGridObject)
  (name string)
  (currentPosY int (init -1))
  (currentPosX int (init -1))
  (teammate bool)
  (carriesGold int (init 0))
  (intention Intention))
```

Listing 1.1. Extract from GoldWorld ontology

In a further iteration, some higher level plans have been designed, embodied into special roles such as Explorer or Transporter. In particular, we created behaviours for finding gold, moving to a certain position, picking up gold, and