WER'04 Panel:

The Role of Requirements Engineering in the Development of Multi-Agent Systems

julio cesar sampaio do prado leite
PUC-Rio, Brasil
www.inf.puc-rio.br/~julio

There has been an increasing number of literature dealing with the software engineering aspect of building "agent oriented software". In principle, agent oriented software is software implemented in platforms in which software pieces behave with a certain level of independence from other software pieces. An important characteristic of agent oriented software is the level of intelligence available for each software piece, or agent.

The concept of agent is usually linked to the following characteristics: autonomy, interactivity, adaptability, persistence and mobility. As such, an agent is a piece of software that needs to be reactive as well as pro-active. Of course, that to implement such software an infrastructure is supposed to be available as to support the implementation of these concepts. One of such infrastructure is, for instance, a programming language, which should provide native operators and operands to support those non-functional characteristics. As of the time of writing, most of the agent-oriented programming languages available do not fully fulfill their goal.

Given that the agent based software will have distinct characteristics from the software usually written, it is reasonable to question what kind of upstream activities would be necessary as to plan for the construction of this type of software. The literature on the subject has been following the path taken by object oriented software development, and there have been several proposals as to adapt object-oriented models and specification languages to deal with the new non-functional requirements imposed by agent-oriented software. Most of literature, however, follows a process with heavy load of documentation for the tasks of definition and specification.

We understand that development of agent oriented software must be grounded on intentional requirements modeling (goal oriented requirements). Focusing on concepts that do mix functional and non-functional concerns upfront is the proper way of making explicit, at the time of definition, the important characteristics of agent based software. On the other hand, we believe that models that handle goals and agents should be directly refined into executable descriptions. Agents identified at the definition level, should be the agents that tries to fulfill the goals also identified at the definition level. Interactivity of agents should be based on the collaboration models captured at the definition level.

One of the major roadblocks to proper software construction is dealing with the several abstraction levels that one needs to navigate to go from conception to implementation. A major advantage of agent oriented software is its anchoring on entities that have responsibilities and "social" obligations. As such, identity from real world actors and virtual pieces of software will help dealing with abstraction levels, and will be fundamental to the idea of going from social nets of actors to attain certain goals to software agents interacting to fulfill these goals.