An Agent Architecture for Telecommunication Applications: Extended Abstract

Murhimanya Clovis Muhugusa
Microcell Labs Inc.
1250 Blvd René-Lévesque West
Suite 400, Montreal, Québec
H3B 4W8 Canada
clovis@labs.microcell.ca

Objectives and Requirements

Our objective is to design an agent architecture which will serve as a reference model for agent implementations. Additionally, we aim at devising an appropriate methodology for the implementation of agents that fit in the proposed agent architecture.

The architecture should be general enough to support different kinds of agents: from purely reactive agents to deliberative utility-oriented agents mainly in the telecommunications domain. Examples of agent applications include: personal agents for mediating interaction between users and services, marketing agents for introducing new services to users, network management agents for monitoring network operation and taking corrective automated actions when network failure is detected, network resources allocation agents for ensuring that users get enough resources for the quality of service they require, pricing agents and various agents for different services available to users such as weather forecast, stock, etc.

All the above agents will act in a complex and non-deterministic environment. In such a setting, robustness is a major requirement. To achieve robustness, agents should be implemented by different modules running in parallel and asynchronously, without any centralized control mechanism. The rationale for this, is the observation that complex and robust systems are inherently decentralized: no single element of the system is crucial for the operation of the whole system. Notable examples of such systems are the Internet, markets, and various societies of insects.

We expect agents to be very complex pieces of software. To master such complexity, agents have to be implemented in a modular and step wise way. Each module should implement a well identified functionality. Modules should be added incrementally to an agent to enhance its capabilities.

Finally, agents will use a combination of different techniques to achieve their goals. For example, to adapt itself to its environment, an agent will use learning. However, no single learning technique is appropriate for all the situations the agent might face. Our approach is therefore to have the agent use different learning techniques implemented by different modules. In circumstances where some technique is not efficient, a different one might be more efficient. With this approach, we expect increased effectiveness from the agent in achieving its goal. Not surprisingly, expert human agents routinely use different approaches to accomplish their tasks. Their expertise is in fact due to their ability to apply appropriately the adequate approach in each circumstance.

The Agent Architecture

Our architecture is a hybrid architecture inspired by the subsumption architecture (Brooks 1986; 1991a; 1991b) and deliberative architectures such as the BDI architecture (Rao & Georgeff 1992; 1995; Kinny, Georgeff, & Rao 1996). The aim is to bring in a single agent architecture the robustness and reactiveness of subsumption-like architectures and the goal-orientedness of deliberative architectures. All the challenge is to marry the advantages of these different approaches without incurring their respective disadvantages.

In order to achieve the requirements presented in the above section, we use 3 parameters to structure an agent: *the mental parameter, the competence parameter*, and *the replication parameter*. The mental parameter characterizes the mental level of the agent. Purely reactive agents work at the *reflex* level. Deliberative agents work at the *cognitive* level and social-aware agents work at the *social* level.

At the cognitive level, we intend to use the BDI model. Finally, we intend to use a market-oriented approach to model the behavior of social-aware agents. Each agent maintains a model of the capabilities and the effectiveness of other agents to solve some kinds of problems. When an agent is faced with a problem it cannot solve, it makes a bid to other agents that it believes are able to help solve the problem. Communication between agents may be based on a subset of KQML (Finin *et al.* 1994) or on a contract net (CNET) based protocol (Smith 1977; 1980; Tidhar & Rosenschein 1992).

The rationale for the competence parameter is to represent an agent as a collection of orthogonal competence modules at the different levels of the mental parameter. This requires the agent implementor to identify clearly

different orthogonal functionalities that will determine the agent behavior. In this way, functionality may be added incrementally to an agent. Finally, the replication parameter reflects the fact that different functional modules of an agent will provide the same functionality using different strategies and techniques.

Prototype implementations will serve to identify functionality needed by different kinds of agents. Such functionality will be implemented by various competence modules that may be reused in the implementation of other agents. In the long term, we expect to have a 'toolkit' of competence modules which can be combined in various ways to build other agents.

To achieve robustness, interaction between the different modules composing an agent must occur asynchronously. We intend to use for this an asynchronous message queueing mechanism. Modules will dynamically subscribe/unsubscribe for messages originating from other modules. This defines a kind of network through which information flows in the agent. In this way, functional modules create dynamic affinity links in the agent. This results in a flexible architecture because the agent's architecture is determined not only by the functional modules that compose the agent, but also by the affinities between them.

High-level mechanisms for coordination, cooperation and conflict resolution among modules will be implemented on top of the asynchronous messaging system. In fact, if one considers agent functional modules as agents with limited capability, the proposed architecture advocates to structure a single agent as a 'multi-agent system'. Like a multi-agent system, the behavior of the agent 'emerges' as the result of its functional modules. And like a multi-agent system, our architecture must address the issues of coordination between functional modules. Dealing with these issues at the agent level will result in more insight and more experience which will be valuable when we will address multi-agent issues. The architecture may therefore be used as a common and unified framework for designing both agents and multi-agent systems.

Implementation Environment

The best way to implement concurrency among functional modules is to implement an agent as a multi-threaded process with each functional module being a separate thread. The multi-threaded approach allows concurrency while providing efficient communication mechanisms between threads.

The development of a complex agent application consists of two related tasks, namely, knowledge engineering and the design and implementation of different processes such as modelling, learning, planning and reasoning, that use the knowledge to achieve rational behavior. The way these processes are implemented depends on the type of knowledge and its representation. Thus, the environment must support different knowledge representation schemes. Furthermore, the environment should allow the agent implementor to engineer the agent knowledge in a modular and step-wise approach similar to that used for the agent's functional modules. Additionally, to support modularity and software reuse, an object-oriented approach is highly desirable.

Finally, support for mobility is interesting due to the foreseen kinds of applications. This requires effective solutions to security issues inherent to agent mobility.

References

Brooks, R. A. 1986. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation* 2(1):14–23.

Brooks, R. A. 1991a. Intelligence Without Reason. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, 569–595.

Brooks, R. A. 1991b. Intelligence Without Representation. *Artificial Intelligence* (47):139–159.

Finin, T.; Fritzon, R.; McKay, D.; and McEntire, R. 1994. KQML as an Agent Communication Language. In *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*. ACM Press.

Kinny, D.; Georgeff, M. P.; and Rao, A. S. 1996. A Methodology and Modelling Technique for Systems of BDI Agents. In van der Velde, W., and Perram, J. W., eds., *Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW'96*, volume 1038 of *Lecture Notes in AI (LNAI)*, 56–71. Springer Verlag.

Rao, A. S., and Georgeff, M. P. 1992. Modelling Rational Agents within a BDI-architecture. In Allen, J.; Fikes, R.; and Sandewall, E., eds., *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*. San Mateo, CA: Morgan Kaufmann.

Rao, A. S., and Georgeff, M. P. 1995. BDI Agents: From Theory to Practice. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*. San Francisco, CA: AAAI Press / The MIT Press.

Smith, R. G. 1977. The contract net: A formalism for the control of distributed problem solving. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)*.

Smith, R. G. 1980. The Contract Net Protocol. *IEEE Transactions on Computers* C-29(12).

Tidhar, G., and Rosenschein, J. 1992. A Contract Net with Consultants. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI-92)*, 219–223.