Open Agent Architecture

About OAA


News!


Applications


Publications

Distribution & Documentation


Contact Information


Back to SRI AI Center

The Open Agent Architecture

A framework for integrating a community of heterogeneous software agents in a distributed environment.

About the Open Agent Architecture

What is an Agent?

The term "agent" has been used by many people to mean many different things. Even within the Agent Research Community, there are at least the following variants on the term *agent*: Mobile Agents (e.g. Telescript), Learning Agents, Autonomous Agents (e.g. robots), Planning Agents, Simulation agents, Distributed Agents.

In the context of the Open Agent Architecture (OAA), we are focused on building distributed *communities* of agents, where *agent* is defined as any software process that meets the conventions of the OAA society. An agent satifies this requirement by registering the services it can provide in an acceptable form, by being able to speak the Interagent Communication Language (ICL), and by sharing functionality common to all OAA agents, such as the ability to install triggers, manage data in certain ways, etc. In our community of agents, we are able to include, and make use of, each of the different types of agents mentioned above.

Agent Architectures as a programming methodology

Distributed Agent technology can be thought of as the next step in the evolution of programming methodologies. *In the beginning*, there were machine and assembly languages. These evolved into higher level programming languages able to break apart programming steps into *subroutines*. A next generalization allowed programmers to group collections of subroutines into *libraries* or

modules. A subsequent innovation added the notion of object orientation: data and routines could be grouped into a single *object*, which further encapsulated the internals of the routines and increased modularity and reuse. Distributed Object technologies, such as CORBA or DCOM, then broke the rule that every object must reside on the local machine; now object libraries could post services through a broker, and the objects themselves could even be written in different programming languages, as long as they used the same Interface Definition Language.

So, what can *Distributed Agents* possibly add to the Distributed Object paradigm? With distributed objects, even though objects may run on different platforms, applications generally form a single monolithic entity of tightly-bound objects, with hand-coded calls to known methods of pre-existing objects.

In a distributed *agent* framework, we conceptualize a dynamic community of agents, where multiple agents contribute services to the community. When external services or information are required by a given agent, instead of calling a known subroutine or asking a specific agent to perform a task, the agent submits a high-level expression describing the needs and attributes of the request to a specialized *Facilitator agent*. The Facilitator agent will make decisions about which agents are available and capable of handling sub-parts of the request, and will manage all agent interactions required to handle the complex query. **The advantage?** Such a distributed agent architecture allows the construction of systems that are more flexible and adaptable than distributed object frameworks. Individual agents can be dynamically added to the community, extending the functionality that the agent community can provide as a whole. The agent system is also able to adapt to available resources in a way that hardcoded distributed objects systems can't.

Human calling Agent, come in Agent...

When designing the Open Agent Architecture, we realized that it is imperative that the human user must be able to interact with the collection of distributed agents as an equal member of the community, not just as an outsider to whom is presented a result once real agents have done all the work. Multiple agents can provide services for retrieval, combination, and management of the growing amount of online information, but this is only useful if controlling and interacting with the network of agents remains less complicated than interacting with the online services themselves!

With this in mind, we designed the InterAgent Communication Language (ICL) to be a logic-based declarative language capable of representing natural language expressions. In addition, we incorporated techniques into the architecture for communicating with agents using simulataneous multiple (natural) input modalities; humans can point, speak, draw, handwrite, or use standard graphical user interface when trying to get a point across to a collection of agents. The agents themselves will compete and cooperate in parallel to translate the user's request into an ICL expression to be handled. These techniques, in combination with the use of special class of agents called Facilitator agents (Facilitator agents reason about the agent interactions necessary for handling a given complex ICL expression), allow human users to closely interact with the ever-changing community of distributed agents.

community of distributed agents.

Technical Features