A Mediated Architecture for Multi-Agent Systems

Gordon Streeter
Sentar
4900 University Square, Suite 8
Huntsville, Alabama 35816

Andrew Potter
Sentar
4900 University Square, Suite 8
Huntsville, Alabama 35816

Tony Flores
Sentar
4900 University Square, Suite 8
Huntsville, Alabama 35816

Abstract:

The technical obstacles to development of the Knowledge Web are formidable. A solution to this problem will include an intelligent agent technology capable of serving limited collections of information resources. It will also provide a scalable infrastructure wherein agents can perform effectively. The prototype architecture uses intelligent software agents to locate relevant knowledge elements and synthesize those elements to produce a usable solution. Individual agents are implemented using knowledge agent shells. These shells are reusable entities tailored to manage specific kinds of information, such as databases, models, and expert systems. Agents utilize managed ontologies and perform knowledge-based reasoning to service information requests. Requests are submitted to the agents via the services of a higher-level entity known as a meta-agent. Meta-agents provide administrative oversight of the problem-solving process, marshalling resources and driving agent interaction towards a solution. The Knowledge Web thus combines centralized ontology-driven agent mediation with decentralized knowledge operations. This combination provides a flexible infrastructure capable of exploiting a variety of distributed knowledge sources while serving a heterogeneous and dynamic user population.

Keywords:

conceptual graphs
diagnosis
knowledge representation
multi-agent systems
ontologies

A Mediated Architecture for Multi-Agent Systems

Abstract:

The technical obstacles to development of the Knowledge Web are formidable. A solution to this problem will include an intelligent agent technology capable of serving limited collections of information resources. It will also provide a scalable infrastructure wherein agents can perform effectively. The prototype architecture uses intelligent software agents to locate relevant knowledge elements and synthesize those elements to produce a usable solution. Individual agents are implemented using knowledge agent shells. These shells are reusable entities tailored to manage specific kinds of information, such as databases, models, and expert systems. Agents utilize managed ontologies and perform knowledge-based reasoning to service information requests. Requests are submitted to the agents via the services of a higher-level entity known as a meta-agent. Meta-agents provide administrative oversight of the problem-solving process, marshalling resources and driving agent interaction towards a solution. The Knowledge Web thus combines centralized ontology-driven agent mediation with decentralized knowledge operations. This combination provides a flexible infrastructure capable of exploiting a variety of distributed knowledge sources while serving a heterogeneous and dynamic user population.

1 Introduction

The vision of a World Wide Web orchestrated by a network of autonomous agents, conspiring to bring order out of chaos, providing the user with precisely the information needed, precisely when it is needed, is an alluring but elusive promise. Technical obstacles to realizing this vision include the unrestricted growth of the Web, the profusion of unmanaged content, and the bewildering variety of rapidly evolving technologies. Although we can imagine harnessing the capabilities of a few web sites, scaling this vision upwards to address the full magnitude of the problem suggests an unmanageable proliferation of autonomous agents. Any solution to this problem will have both an intelligent agent technology that can serve limited collections of information resources, and scalable infrastructure wherein these agents can perform effectively.

This is the basis of our approach to the design of the Knowledge Web, as presented in this paper. This Knowledge Web, called the KnoWeb, uses intelligent software agents to locate relevant knowledge elements and synthesize them to produce a usable information resource. Individual agents are implemented using knowledge agent shells. These shells are reusable entities tailored to manage specific kinds of information, such as databases, models, and expert systems. Agents utilize managed ontologies and perform knowledge-based reasoning to service information requests. Requests are submitted to the agents via the services of a higher-level entity known as a meta-agent. Meta-agents provide administrative oversight of the problem-solving process, marshalling resources and driving agent interaction towards a solution. The Knowledge Web thus combines centralized ontology-driven agent mediation with decentralized knowledge operations. This combination provides a flexible infrastructure capable of exploiting a variety of distributed knowledge sources while serving a heterogeneous and dynamic user population.

2 Shared Knowledge Representation

We use conceptual graphs for both ontology and knowledge representation. A benefit of this is that knowledge and its corresponding ontology are conceptually isomorphic. To support programmatic transmission and manipulation, these graphs are implemented in XML, but may also be rendered in KL0, a notation we have developed that is similar to the conceptual graph linear form of representation [Sowa, 2000]. Figure 1 shows an example of this bi-level representation.

```
(Between)
    +-inner--> [ Person: ]
    +-outer--> [ Rock: ]
    +-outer--> [ HardPlace: ]
```

```
<relation type="Between">
  <arc label="inner">
    <concept type="Person"/>
  </arc>
  <arc label="outer">
    <concept type="Rock"/>
  </arc>
  <arc label="outer">
    <concept type="HardPlace"/>
  </arc>
</relation>
```

Figure 1 - Bi-level Knowledge Representation

All knowledge expressions are conformant to an XML DTD that provides a mechanism for describing conceptual graphs. Each agent's knowledge module uses an ontology. Ontologies are constructed of three sections:

- Logical, including primitives such as proposition, negation, and disjunction.
- Task, consisting of general concepts such as cause, symptom, and diagnosis
- Domain, consisting of factual and heuristic knowledge of the subject specialization

Although our current work is based on a diagnostic paradigm, other paradigms are presumably possible. Providing a common syntax for all knowledge representation enables meta-agents to manipulate domain content without requiring visibility into content. Since expressions are ontologically constrained, meta-agents can collect knowledge from a variety of sources and perform knowledge operations on them while limiting domain dependency.

3 Agent Technology

Not all agents are equal. Some are specialists, focusing on a narrow domain. Others are generalists, capable of synthesizing information from a variety of specialized sources. Still others possess knowledge of other agents. Each kind of agent brings unique and essential capabilities to the Web:

- *Knowledge agents* have capabilities within a subject specialization.
- *Domain Advisor Agents* support meta-agents, working as knowledge advisors. Met-agents use advisor agents to help plan knowledge transactions, manage conflict resolutions, and direct activities toward an attainable objective.
- *Service Agents* provide an agent capability brokerage. A Service Agent matches requests with capabilities.
- *User Agents* specialize in communication with a user.
- *Meta-agents* perform agent mediation during knowledge transactions, marshalling resources and managing agent interaction.

More detail on each agent is provided later in this paper. Figure 2 shows an example of how these agents interrelate.

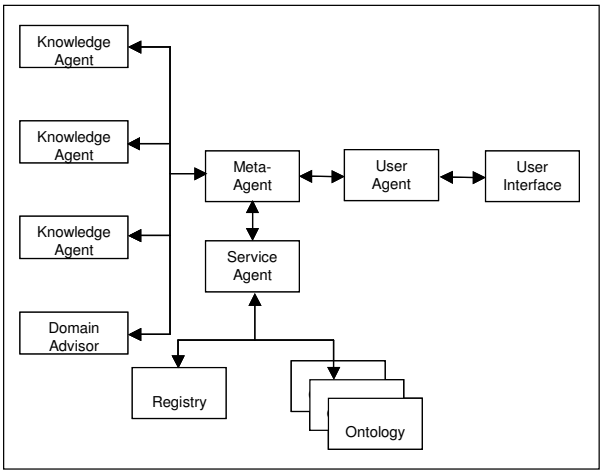

Figure 2 - Multi-Agent Knowledge Web Architecture

3.1 Knowledge Agents

Knowledge agents focus their capabilities within a specialized domain. Each agent publishes its capabilities by registering with a service agent.

Knowledge agents are implemented using a reusable symbolic logic shell. This shell, shown in Figure 3, has the ability to initialize the underlying logic engine and the agent communication subsystem, and performs translation on requests and responses. The symbolic encoding used by the logic engine is defined by the internal processing requirements of the particular engine.

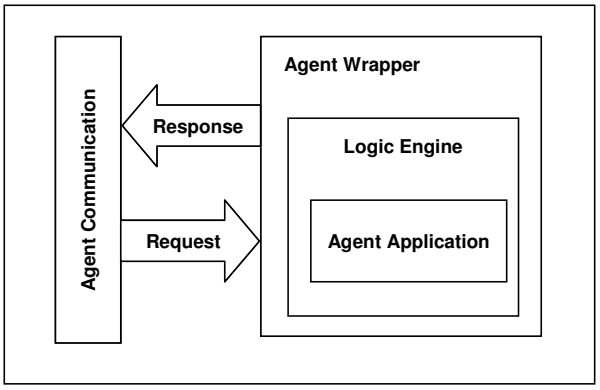

Figure 3 - Symbolic Logic Agent Shell Functional Components

The symbolic logic shell supports migration of legacy resources to the Knowledge Web architecture. It has been

adapted to support several resource types. We have used it to create a FAQ agent, a service agent, and an advisor agent.

3.2 Domain Advisor Agents

Advisor Agents are knowledge agents that provide domain dependent support to meta-agents, working as application knowledge advisors. Our current implementation is an expert system agent using the symbolic logic agent shell.

When an advisor agent accepts an inquiry from a meta-agent, it first checks its knowledge context for an answer. If the answer is available, it sends it to the meta-agent. If no response is readily available, the advisor forms its own inquiry and attempts to develop an answer by drawing upon its internal knowledge resources as well as those of other agents.

3.3 Service Agents

Service agents manage the published capabilities of specialist agents. When presented with a request, the meta-agent calls upon the service agent to identify the appropriate destination agents for the request. The destination agent will usually be a knowledge agent, representing a unique domain specialization. The meta-agent dispatches the question to the destination knowledge agent. In the process of creating a response, the knowledge agent may need to post queries of its own to the system. The meta-agent again uses the service agent to find a suitable destination agent and dispatches the question accordingly.

3.4 User Agents

User agents are knowledge resources for user interfaces, and they are intermediaries between meta-agents and users. The user agent accepts questions from the user. Before forwarding them to the meta-agent, it ensures that they are ontologically well formed and potentially answerable. Responses received from the meta-agent are rendered from KL0 to presentation format for the user.

3.5 Meta-Agents

The task of a meta-agent is to enlist and mediate between knowledge agents as necessary to achieve a goal. By providing administrative oversight to multi-agent operations, the meta-agent performs high-level reasoning, or reasoning *about* the reasoning capabilities of others. The meta-agent orchestrates the activities of other agents, exploiting their individual capabilities and ensuring the integrity of their responses.

When a meta-agent initiates a knowledge transaction, it consults a service agent for the identity of useful knowledge agents. During the course of the transaction, additional agents are called into play as needed. Alliances are created as needed and just as quickly discarded. There is no static, fixed architecture. There are only the opportunistic relationships in use at any given moment during a knowledge transaction.

The meta-agent uses an agenda to keep track of what it is doing and why. Broadly taken, this would include the entirety of the meta-agent code, but the more significant agenda elements include:

- **Goal.** Representing an inquiry under consideration, the goal is the highest-level structure of the agenda.
- **Context.** A collection of propositions asserted along a particular path through an agenda
- **Prospect.** Each registry entry applicable to a goal is represented by a prospect.
- **Evocation.** An evocation is a record of a request sent to a Knowledge Agent and all processing that arises from this transaction.
- **Postulate.** The agenda equivalent of a proposition, the postulate includes the proposition and information as to its origin.

When the meta-agent opens a new context, its first activity is to identify a domain advisor and form an alliance with it. A domain advisor is an agent claiming the ability to provide assistance in the domain-specific aspects of problem solving. The meta-agent identifies the domain advisor by asking the service agent which agent, if any, it should use. By handling this step as an inquiry, the meta-agent is able to avail itself of the same agenda management, service agent look-up, and conflict resolution services that are provided as a matter of course in any inquiry.

The meta-agent then performs marshalling activities to identify and distill the list of knowledge agents to be consulted. Marshalling consists of identifying and selecting applicable registry entries. This is accomplished using the following processes:

- **Identify.** The meta-agent interacts with the Service Agent to identify the registry entries of Knowledge Agents that may have application to the task at hand.
- **Preclude.** The meta-agent discards any registry entries with preconditions that contradict the context. This is accomplished with assistance from the domain advisor.
- **Consider.** The meta-agent evaluates any preconditions associated with the remaining registry entries. It does this by creating

inquiries for each precondition and submitting these inquiries to itself.

- **Exclude.** After each precondition is considered, further conflict resolution must be performed. The meta-agent excludes all entries whose precondition is contradictory to the context. This is accomplished with assistance from the domain advisor.

Once marshaling is complete, the meta-agent dispatches the inquiry to the identified agents and waits for these agents to reply. If the condition of any inquiry yields more than on value, a separate branch must be created for each value. Each alternative is addressed independently. Conceptually, this is accomplished by branching the agenda and providing each branch with a separate copy of the context. During the interval between request and the response, any of the marshaled agents may submit new inquiries which must be processed before the evaluation is complete.

Once all solutions are received, the Meta-agent reduces the solution set to unique solutions by arbitrarily discarding duplicate entries. If a domain advisor is available for the context, it is given the opportunity to discard solutions for some domain-dependent reason.
The Meta-agent has no concept of truth. What agents wish to assert, the Meta-agent accepts as postulate. This nature of speculative credulity leaves the Meta-agent open to contradiction, and though cavalier with truth, it is a stickler for consistency. The Meta-agent tries first to prevent contradiction, and then to resolve it, but it is prepared, finally, to handle it with a blind meticulousness. As the Meta-agent is unaware of the real impact of and contradiction, it must isolate each contradictory postulate in its own copy of the context, protecting the consequence of any one from the influence of any other. Each such postulate is called the "antecedent" of the precedence. Likewise, while in other systems this situation might be characterized as "uncertainty" and be dealt with in terms of "possibility" or "probability", to the Meta-agent these separate lines of inference are simply "alternatives".

The Meta-agent concurrently processes multiple requests and reentrantly processes requests that arise within the context of other requests. Its implementation is event-based. Incoming messages are events.

4 Conclusion

KnoWeb technology provides the means for managing the Knowledge Web, while permitting it to thrive as a vast collection of autonomous resources. Whereas the populist nature of the Web as we now know it seems resolutely anti-architectural, the mediated architecture described here not merely overcomes this impediment, but draws upon it as a positive feature.

Service agents provide the means for agents to publicize their capabilities, and Meta-agents, with assistance from Advisor Agents, provide the mechanism for managing diverse resources, resolving conflicts, and synthesizing responses. By adhering to a consistent knowledge representation scheme based on an industry standard mark-up language, the KnoWeb provides a common language that may be understood and transmitted by diverse agents. Provision of the symbolic logic agent shell supports adoption of legacy web resources into the architecture.

Future iterations of the prototype call for development of enhanced explanation capabilities and an intelligent user interface. A prototype knowledge builder is currently under development. This will enable creation of larger knowledge modules, which will support further testing and demonstration.

References

[Sowa, 2000] John F. Sowa, *Knowledge Representation; Logical, Philosophical, and Computational Foundation,* Brooks/Cole, Pacific Grove, California 2000