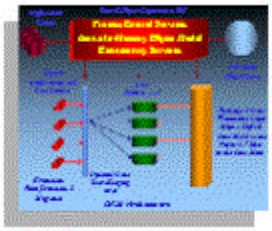

DKMS Briefs

Joseph M. Firestone

DKMS Brief No. Three: Software Agents in Distributed Knowledge Management Systems

Agents and DKM Architecture

In recent White Papers [1] and in DKMS Brief No. One [2], I presented Distributed Knowledge Management (DKM) architecture as the appropriate architecture for Distributed Knowledge Management Systems (DKMS), and DKMS as the IT application whose purpose is to support the Knowledge Management Process. An important aspect of DKM architecture is its Active Knowledge Manager (AKM) component.

The AKM provides process control/distribution services, an in-memory active object model accompanied by a persistent object store, and connectivity to a variety of data stores and application types. One way the AKM can perform some of its process control services is through software agents. Let's first briefly review some fundamentals on software agents, and then see how agents can contribute to process control services.

An Overview of Software Agents

A Software Agent (SA) is a software object that acts on behalf of another software object (its client) and behaves to at least some degree: autonomously (without continuous direction), socially (interacts with other agents), proactively (influences its environment), and reactively (is influenced by its environment). [3] An intelligent software agent is an SA that: has an in-memory knowledge base including cognitions, evaluations, goals, and perhaps even affects; is rational in the sense that it makes decisions, acts to attain its goals; and learns.

A static SA is one that does not move from the platform that creates it. A Mobile SA [4] can move across a network from one physical computer to another. It can do this autonomously, as it perceives the need for such movement. It takes its run-time environment with it wherever it goes. It can stop program execution on one computer, move to another computer and then begin again at the second computer, interacting with that computer to communicate and/or gather data, information, or knowledge.

Contrast the mobile SA concept with the original client/server model. In the client/server model, a single request is sent over a network and activates a computing procedure at the destination computer. A result is then sent across the network to the client. In contrast, a mobile SA travels to a server and then may perform a variety of transactions with it. Eventually, when its business with the destination computer is done, it either returns to the source computer with the results of its transactions, or moves to another destination computer to transact still more business.

The "source computer" of a mobile SA is its home agency. [5] The agency consists of a computing

environment, an agent scripting capability, and a database. Mobile SAs register with home agencies. They also register as visitors with other agencies. Some Mobile SAs are Broker Agents. They recruit other agents to create task forces and delegate work to the agents they recruit. They can also contract with other agents as part of the recruitment process.

Mobile SAs and their agencies require a host environment in order to execute. This is a distributed computing environment overlaying a host distributed computing environment. It provides various essential services to mobile SAs, including the ability to create them, and the ability to execute. [6]

Software Agents and Process Control Services

The DKMS is a system providing support for business processes composed of Planning, Acting, Monitoring, and Evaluating sub-processes [7]. In turn, these sub-processes are composed of more specific sub-processes or use cases, and these, in turn, are composed of tasks. A comprehensive analysis of how agents can participate in process control services, would need to go through the details of how each use case and task sub-process might be performed by agents and the DKMS's agent-based infrastructure. That's way beyond the scope of this column. What I will do though, is to provide a general characterization of how SAs fit into process control services, and leave detailed analyses of how they contribute to individual use cases for future Briefs and White Papers.

Process Control Services in the DKM include:

- in - memory proactive object state management and synchronization across distributed objects;
- component management;
- use case and workflow management; and
- transactional multithreading.

Agents have no role in transactional multithreading, which is a fundamental aspect of the AKM. But they may contribute, as DKM architecture evolves, to the other three areas.

Object/Component Management and Agents

In the DKMS, business objects will be shared across data warehouse and data mart applications, and will be stored in an in-memory object model. The DKMS through the AKM must have the ability to monitor and coordinate changes in the shared classes and objects across these applications and across their different physical platforms. This means the ability to monitor and coordinate changes in attributes and methods of the shared objects automatically. Let's call this ability Dynamic Integration [8]. To perform dynamic integration, the system must:

- look for changes in shared objects and additions to the total pool of objects and relationships,
- alert all system components sharing the objects of such changes, and also
- make decisions about which changes should be implemented in each affected component throughout the system.

It is important that changes in shared objects are propagated and new objects are created in real-time, so that a single view of the DKMS object model is maintained. This is why in-memory, proactive operation is so important.

Like objects, components can also be shared across applications and physical platforms. Component management is the ability to monitor, coordinate, and synchronize changes in components, and is analogous to object state management at the component level. It too, needs to be performed in real-time, and it too requires

proactive, in-memory operation to be most effective.

Agents can play a major role in performing dynamic integration as part of the AKM. The AKM is itself composed of distributed object models, made up of reflexive objects [9]. A reflexive object is one that is aware of changes in its state. When a change is introduced in one of these objects it communicates the change (through an alert) to a central object model within the AKM. The central object model contains a view of all objects and relationships in the DKMS. The central object model will respond to this alert by incorporating the changes into the central object model and deleting the old versions of the objects, as long as no other object models share the old object versions. If they do, the central object model will dispatch Negotiator mobile agents to the various distributed object models incorporating old object versions.

The task of these Negotiator mobile agents is to negotiate with the effected distributed object models about whether the changed objects are acceptable to them. The distributed object models can employ static agents to negotiate for them. If the changed objects are acceptable, the old versions of the objects can be deleted from all object models, and the new objects can be incorporated into all distributed object models. If not, the central object model will maintain both the old and the changed objects to accommodate disagreements among the distributed applications.

Both the mobile and the static agents involved in the mutual coordination process will need some intelligence. That is, they will exhibit cognitions, evaluations, and goals, will make decisions, and perhaps should have the capacity to learn from previous negotiations with other agents.

Why are negotiator agents desirable in performing dynamic integration? While dynamic integration can be performed without negotiator agents, the advantage in using them comes from better performance. Without negotiator agents all of the transactions in negotiations between central and local components of the AKM would flow over the enterprise network. With them, only the agents are sent from the central AKM component to other components. Negotiations actually occur on the target rather than the source platform. When the agent returns to the central AKM component, it brings back only the result of the negotiation.

Use Case/Workflow Management and Agents

DKMSs support business processes by assisting efforts to gather, organize, create, maintain, and enhance knowledge about them, and also by providing support for planning, implementing, monitoring, and evaluating the course of the business process. Both use cases and work flows are task sequences within these activities that process, route, and distribute information products, but the connotations of the two terms are somewhat different. The use case concept looks at a task sequence from the point of view of the valued outcome the user will get from a task sequence. Work flow, on the other hand, refers to the automated system constructed to implement a use case, a part of a use case, or a set of related use cases. Process control services must provide the means to manage such work flows by:

1. facilitating specification of routing and distribution;
2. supporting rapid and easy change in the routing structure, the distribution process, and the business rules governing the work flow;
3. providing the capability to either store the product of a work flow task or "push" it to the next step in the work flow;
4. providing the capability to distribute the work flow process across multiple computers;
5. providing the capability to gather knowledge resources to support the work flow;
6. supporting collaborative transactions among work flow participants;
7. providing the capability to simulate the work flow; and
8. providing the capability to customize work flows by integrating custom, legacy, or external data and/or applications.

Agents may be applied in work flow process control areas 3-8.

Area 3: In deciding whether to store the product of a work flow task or push it to the next step, negotiator agents of the components performing the steps can exchange information on the depth of their work queues; and on their relative abilities to store and process the next step in the work flow. Together they can decide on whether the work flow item in question will be stored or "pushed." In case of disagreement the central AKM component can arbitrate.

Area 4: Agents based at each component, can also increase the capability to distribute the work flow process by continuously monitoring their components and alerting the central AKM component if processing capability is stressed [10]. The central AKM agent can then assist the "local" agents in negotiations to distribute the work load.

Area 5: Knowledge Retrieval agents, next, can help in providing the capability to gather knowledge resources to support a work flow. Such agents can model [11] each individual information or knowledge resource within the DKMS. They can then collaborate with Interface agents, receiving queries from them and transmitting only the results to the interface agents. Various types of knowledge may be retrieved by such agents including descriptive, impact-related, predictive, outcome assessment, and benefit/cost assessment knowledge.

Area 6: Intelligent Interface agents can support collaborative work flow activity in useful ways. For example, in planning, a number of decision makers may have to agree on a hierarchy of goals and objectives, and ultimately on a planning option. Interface agents can help planners to be explicit about the goals, objectives and priorities that comprise their planning hierarchies. Then negotiating agents for different planners can work together to analyze the similarities and differences in planning hierarchies and to negotiate a common planning hierarchy.

Interface agents and negotiating agents can also be important in developing concrete planning options incorporating planning hierarchies and action effect scenarios into plans. Planners will differ not only in their planning hierarchies, but also in their cognitive maps relating actions and effects. Again, interface agents can help planners be explicit about their cognitive maps, and negotiating agents can work together to arrive at a common cognitive map underlying a preferred planning option.

In addition to supporting planning, interface and negotiating agents can support collaborative work in Knowledge Discovery in Databases (KDD) activity [12]. Here analysts will disagree on both cognitive maps expressed in formal models, and on validation criteria used to select among models. Interface agents can help analysts to perform formal modeling, they can also help them in formulating their validation schema supporting model choice. Negotiating agents can then assist analysts in arriving at common validation schema.

Area 7: Agents can also assist in simulating work flow systems. Systems can be represented by agents functioning as the nodes of a work flow. Agents can be assigned tasks they perform according to rules programmed in the agents and triggered by events and their parameters. Work flow items can be defined to provide agents something to process. When the simulation is run various characteristics of the work flow design can be evaluated.

Area 8: Agents provide only one way to integrate custom, legacy, or external data and applications into a work flow system. But agent technology can be used to produce a simple information agent by "wrapping" any information source to allow it to conform to the communication conventions of an agent infrastructure [13]. While this is not so much a contribution to process control in itself, it does support other agents in the DKMS infrastructure by facilitating communications between such simple information agents and other more proactive agents, and by providing a capability to script the onformation agents to perform simple functions such as scheduled reporting and alerting of other agents to important events reflected in the information source.

Conclusion

Software agents are among the most significant technical developments in IT today. In previous columns and White Papers on the DKMS and on DKM architecture I have not been explicit about the role of agents in the DKMS, but instead have concentrated more generally on distributed objects and components. In this column I've provided a general viewpoint on how agents fit into the DKMS picture. Most generally, I've claimed that they can be vital in solving object and component state and change management, and in solving the Dynamic Integration Problem. In addition, I've pointed to a wide range of contributions agents can make in supporting various aspects of the use cases and associated work flows comprising the DKMS.

In future columns I'll take up agents again. But this time the focus will be on specific use cases in the DKMS and on associated work flows. By exploring the role of agents in this concrete way I will begin to fill in some of the details of DKM architecture.

References

[1] I introduced the DKMS concept in two previous White Papers "Object-Oriented Data Warehouse," and "Distributed Knowledge Management Systems: The Next Wave in DSS." DKM architecture was introduced in a third White Paper "Architectural Evolution in Data Warehousing." All three are available at http://www.dkms.com/White_Papers.htm.

[2] See DKMS Brief No. One: The Corporate Information Factory or the Corporate Knowledge Factory?" at http://www.dkms.com/White_Papers.htm.

[3] There's a very sizable literature dealing with the definition and conceptualization of software agents. Here I've relied on Elizabeth A. Kendall, Margaret T. Malkoun and Chong Jiang, "A Methodology for Developing Agent Based Systems for Enterprise Integration, Royal Melbourne Institute of Technology, available at: http://www.cse.rmit.edu.au/~rdsek/, and Shaw Green, Leon Hurst, Brenda Nangle, Dr. Padraig Cunningham, Fergal Summers, and Dr. Richard Evans, "Software Agents: A Review," Trinity College, Dublin, and Broadcom Eirann Research Ltd., May 27, 1997, available at:

[4 See Robert Orfali, Dan Harkey and Jeri Edwards, The Essential Distributed Objects Survival Guide (New York: John Wiley & Sons, 1998) Pp. 255-256, and 401-405

[5] Ibid.

[6] Op. cit "Software Agents . . ." Pp. 26-27

[7] See op. cit., Joseph M. Firestone, "Distributed Knowledge Management Systems . . ." Pp. 7-15.

[8] The Dynamic Integration Problem is discussed in more detail in "Architectural Evolution . . ." op. cit.

[9] Reflexive objects are used in Template Software Enterprise Integration Template product. See Template Software, "Integration Solutions for the Real-Time Enterprise: EIT - Enterprise Integration Template," Dulles, VA, White Paper May 8, 1998, P. 17 at http://www.template.com.

[10] The Enterprise Integration Template provides a distributed object model as well as process control and connectivity services useful in developing a distributed AKM. See Ibid. Two other products that could be used

to develop an AKM component are DAMAN's InfoManager (inquire at http://www.damanconsulting.com), and Ibex's DAWN workflow product along with its ITASCA active database (at http://www.ibex.ch/)

[11] Information Retrieval Agents similar to what I've called knowledge retrieval agents are conceptualized in "Software Agents . . ." Pp.11-12. I've relied on this development in my treatment.

[12] The Perform KDD use case is described in detail in my "Knowledge Management Metrics Development: A Technical Approach," at http://www.dkms.com/White_Papers.htm.

[13] See Jeffrey M. Bradshaw (ed.), Software Agents (Cambridge, MA: AAAI Press/M.I.T. Press, 1998), P. 31.

Biography

Joseph M. Firestone is an independent Information Technology consultant working in the areas of Decision Support (especially Data Marts and Data Mining), Business Process Reengineering and Database Marketing. He formulated and is developing the idea of Market Systems Reengineering (MSR). In addition, he is developing an integrated data mining approach incorporating a fair comparison methodology for evaluating data mining results. Finally, he is formulating the concept of Distributed Knowledge Management Systems (DKMS) as an organizing framework for the next business "killer app." You can e-mail Joe at eisai@home.com

---

[Up] [KMBenefitEstimation.PDF] [MethodologyKIv1n2.pdf] [EKPwtawtdKI11.pdf]
[KMFAMrev1.PDF] [EKPebussol1.PDF] [The EKP Revisited]
[Information on "Approaching Enterprise Information Portals"]
[Benefits of Enterprise Information Portals and Corporate Goals]
[Defining the Enterprise Information Portal]
[Enterprise Integration, Data Federation And The DKMS: A Commentary]
[Enterprise Information Portals and Enterprise Knowledge Portals]
[The Metaprise, The AKMS, and The EKP] [The KBMS and Knowledge Warehouse]
[The AKM Standard]
[Business Process Engines in Distributed Knowledge Management Systems]
[Software Agents in Distributed Knowledge Management Systems]
[Prophecy: META Group and the Future of Knowledge Management]
[Accelerating Innovation and KM Impact]
[Enterprise Knowledge Management Modeling and the DKMS]
[Knowledge Management Metrics Development]
[Basic Concepts of Knowledge Management]
[Distributed Knowledge Management Systems (DKMS): The Next Wave in DSS]