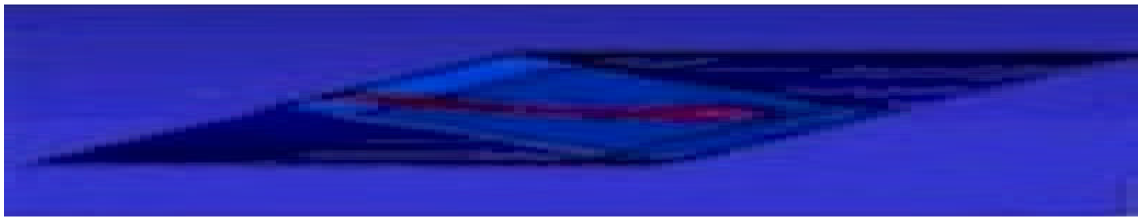

Working Paper No. One

The Artificial Knowledge Manager Standard: A "Strawman"

By

Joseph M. Firestone, Ph.D.

Executive Information Systems, Inc.

http://www.dkms.com

eisai@home.com

Revised March 16, 2000

Prepared for First KMC/AIIM

KM ANSI/ISO Standards Committee Meeting

January 29, 1999

This paper is a working paper, or "straw man," circulated for purposes of collaboration within the Knowledge Management Consortium International's (KMCI) Artificial Knowledge Management Systems Committee (AKMSC). It is intended that this paper be used by the Committee, along with contributions of other committee members to arrive at a collaborative Standard Recommended Practice on Artificial Knowledge Base Management Systems, a product of the Committee and the KMCI.

Introduction

An Enterprise Knowledge Management Model (EKM) is a hierarchical network of rules that enables an agent to explain, anticipate and predict events and interaction patterns: (a) in the enterprise's Knowledge (Kn) Processes, or Knowledge Management (KM) Processes; and (b) in the enterprise's environment. An EKM model represents or models the Natural Knowledge Management System (NKMS) [1] of an enterprise.

An Enterprise NKMS is an on-going, conceptually distinct unit composed of enterprise organizational and human components and their persistent interactions, both having properties, whose interaction properties are not determined by design, but instead emerge from the dynamics of the enterprise interaction process itself. An Enterprise NKMS includes mechanical and electrical organizational components produced by it, such as computers and computer networks, as well as human and organizational agents.

An enterprise Artificial Knowledge Management System (AKMS) is an enterprise wide conceptually distinct integrated component produced by its NKMS:

- whose components are computers, software, networks, electronic components, etc.,
- whose components and basic interaction properties are determined by design, and
- whose overall purpose is to support the Kn and KM processes of the NKMS.

A key aspect in defining the AKMS is that both its components and basic interaction properties must be designed. The idea of being fully designed vs. being partly designed, or not-designed is essential in distinguishing the artificial from the natural. Thus, in an enterprise or any other organization, even though we may try to design its processes, our capacity to design is limited by the fact that it is a Complex Adaptive System (cas) [2] [3] [4]. If we understand a cas, we expect to be able to correctly design its components and certain aspects of its processes, but in the end we expect its behavior to be emergent and to not be a precise consequence of our design. Once the cas starts operating, it is self-organizing and controls its own behavior. At best, we can only interact with it and influence it.

On the other hand, with an AKMS, we design both its components and their basic interactions. The connection between the design and the final result is determinate and not emergent. When we interact with the AKMS, we can precisely predict what its response will be.

The AKMS is designed to manage the integration of computer hardware, software, and networking objects/components into a functioning whole, supporting enterprise knowledge production, acquisition, and transmission processes. The AKMS, in other words, supports producing, acquiring, and communicating the enterprise's knowledge base. The enterprise's knowledge base, in turn, is used by its agents to perform Kn, KM, and business processes. [5] This Working

Paper will present a conceptual model of the AKMS and of its key integrative component, the Artificial Knowledge Manager (AKM) [6]. It is intended as a "straw man" to help the Artificial Knowledge Management Systems Committee begin work and focus its deliberation on an AKM standard.

The AKMS Architecture

Two ways to look at the AKMS are in terms of its use cases and its architecture. After a few words of background on the relation of use cases to architecture, let's proceed to examine AKMS architecture and then its use cases.

In the Unified Modeling Language (UML) a use case is defined as "a set of sequences of actions a system performs that yield an observable result of value to a particular actor." An actor is a human agent. When an AKMS is viewed functionally as an application, its users perform a set of use cases supporting various tasks within the main activities of the knowledge and KM processes of an NKMS. An AKMS doesn't automate all NKMS activities. Only some. Figure One shows the abstract relationship of AKMS Use Cases to knowledge and KM processes

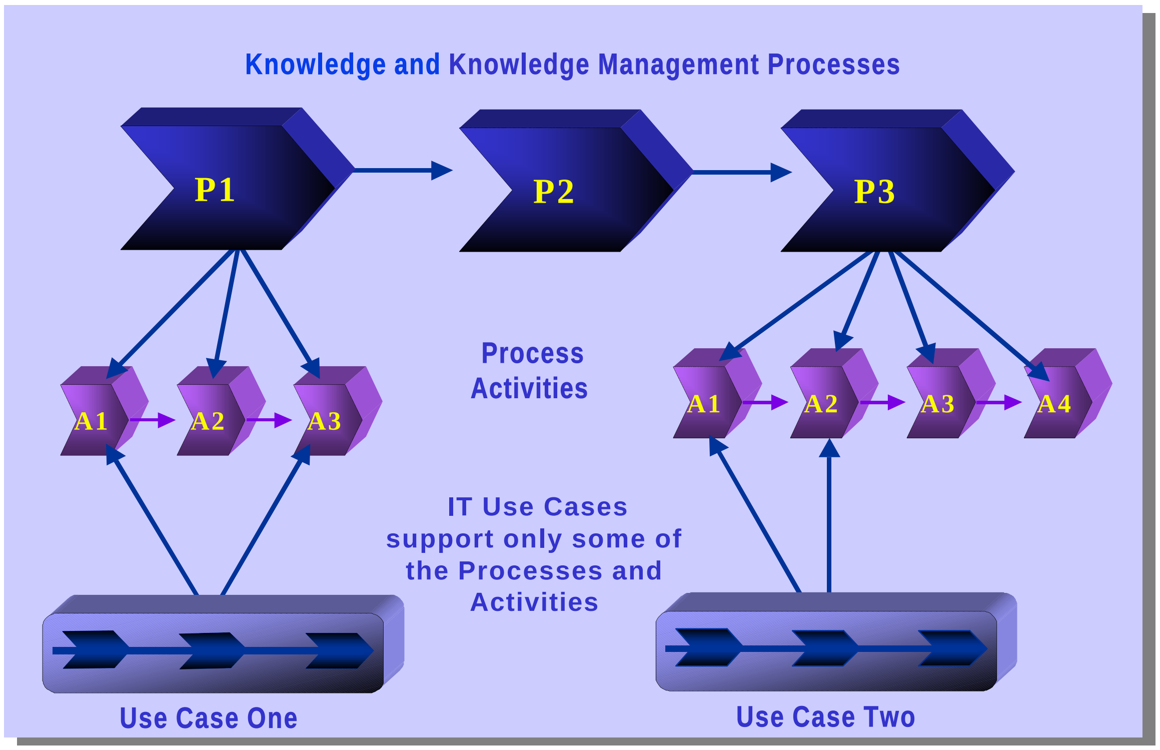
Knowledge and Knowledge Management Processes
P1
P2
P3
A1
A2
A3
Process Activities
A1
A2
A3
A4
IT Use Cases support only some of the Processes and Activities
Use Case One
Use Case Two

Figure One -- Relationships of KM Processes and Activities to AKMS Use Cases

Architectural Overview

If use cases specify the functional or activity aspect of the AKMS, the objects and components of the AKMS that support these use cases, and their interrelationships, provide its structure. We can begin to understand AKMS structure by visualizing a basic, abstract architecture. That architecture is expressed in Figure Two.

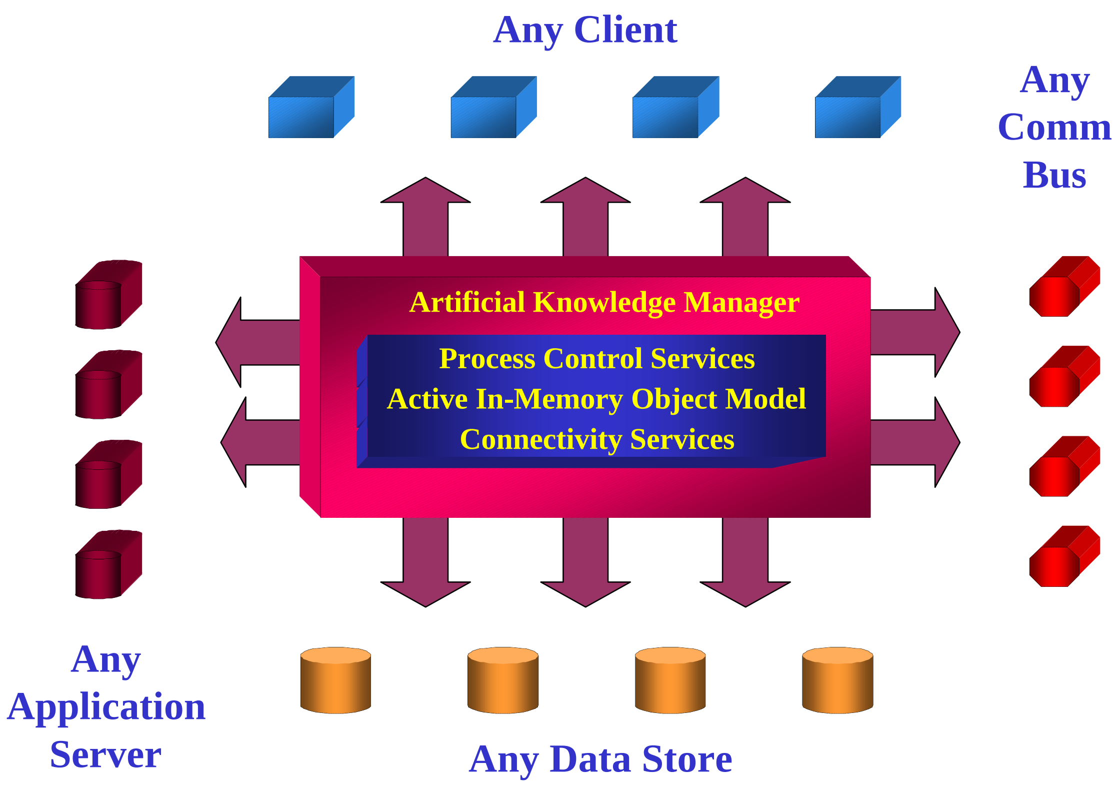

Figure Two -- KMCI AKMS "Straw Man" Architecture

The figure shows clients, application servers, communication buses and data stores integrated through a single logical component called an Artificial Knowledge Manager (AKM). The AKM performs its central integrative functions by providing process control and distribution services, an Active, In-memory Object Model supplemented by a persistent object store, and Connectivity Services to provide for passing data, information, and knowledge from one component to another. I will specify the AKM in much more detail below. For now a more concrete visual picture showing the variety of component types in the AKMS, is given in figure Three.

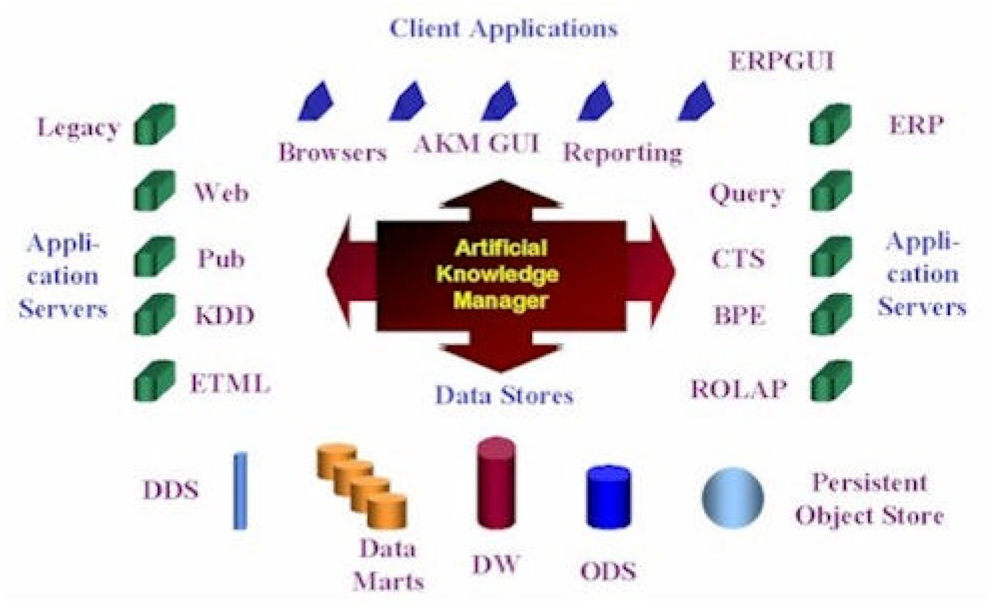

Figure Three -- The AKM, Data Stores,

Applications Servers and Clients

An important difference between the two figures is that the communications bus aspect of the AKMS is implicit in figure Three, where I've assumed that the AKM incorporates it. Figure Three makes clear the diversity of component types in the AKMS. It is because of this diversity and its rapid rate of growth in the last few years that the AKM is necessary.

Change in the AKMS can be introduced through so many sources that if the AKMS is to adapt to change it needs an integrative component like the AKM to play the major role in its integration and adaptation.

The Key Architectural Components of the AKMS are:

- The Artificial Knowledge Manager (AKM)
- Stateless Application Servers
- Application Servers that maintain State
- Object/Data Stores
- Object Request Brokers (e.g., CORBA, DCOM)
- Client Application Components

More detail on these follows.

The AKM [7]

An AKM provides Process Control Services, an Object Model of the Artificial Knowledge Management System (AKMS) (the system corresponding to the AKMS architecture), and connectivity to all enterprise information, data stores, and applications

Process Control Services include:

- In -memory proactive object state management and synchronization across distributed objects and through intelligent agents;
- Component management and Workflow Management through intelligent agents
- Transactional multithreading;
- business rule management and processing; and
- metadata management.

An In-memory Active Object Model/Persistent Object Store is characterized by:

- Event-driven behavior;
- AKMS-wide model with shared representation;
- Declarative business rules;
- Caching along with partial instantiation of objects;
- A Persistent Object Store for the AKM;
- Reflexive Objects.

Connectivity Services should have:

- Language APIs: C, C++, Java, CORBA, COM;
- Databases: Relational, ODBC, OODBMS, hierarchical, network, flat file, etc.;
- Wrapper connectivity for application software: custom, CORBA, or COM-based; and
- Applications connectivity including all the categories mentioned in Figure Three above,

whether these are mainframe, server, or desktop - based.

In the following paragraphs, I'll expand on Process Control Services and the Active Object Model.

Process Control Services

- Object Management and Synchronization

The AKMS supports a variety of data stores and application servers that allow batch, transaction, and DSS processing to occur in the same system. The result of this diversity of processing activities is to introduce frequent and rapid changes into the AKMS, its data stores, and its application servers. Change in data, methods (including business rules), and behavior is the "law of life" in the AKMS.

The problem of managing, synchronizing and adapting to these changes in the AKMS is the Dynamic Integration Problem (DIP). A primary function of the AKMS, and its AKM integrative component, is to automate Dynamic Integration (DI) as much as practicable. To perform dynamic integration, the AKM 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.

The AKM accomplishes these tasks by using its in-memory, shared, active object model with its support for event-driven behavior, a common view of the system's objects, declarative business rules, and caching of data along with use of partial instantiation of objects.

In addition, the AKM relies on a persistent representation of the object model. The objects in the object model are reflexive -- aware of their present state and any change of state [8].

The AKM accomplishes proactive monitoring and coordinating of changes in its shared objects through their reflexivity and capacity for event-driven behavior, and through software agents. The capacity for event-driven behavior causes the objects to adjust in response to event-induced changes in some shared objects by making corresponding changes in themselves. A particular type of AKM event-driven object that is also autonomous is a software agent.

Agents can play a major role in performing dynamic integration as part of the AKM. When a change is introduced in an AKM object, the object 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 AKMS. 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 objects incorporating old object versions.

The task of these Negotiator Mobile Agents is to negotiate with the effected distributed object sub-models about whether the changed objects are acceptable to them. The distributed object sub-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 sub-models, and the new objects can be incorporated into all distributed object sub-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.

- Component Management and Synchronization

Like objects, components can also be shared across applications and physical platforms. And they also change frequently and rapidly and require DI. Component management is the ability to monitor, co-ordinate, and synchronize changes in components, and is analogous to object state management. It too, needs to be performed in real-time, and it too requires proactive, in-memory operation to be most effective.

Component Management and Synchronization in the AKM requires much the same set of capabilities as object state management and synchronization, and can benefit from the use of software agents..

- Work Flow Management

The AKMS supports 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 sequence constructed to implement a use case, a part of a use case, or a set of related use cases. AKM process control services must provide the means to manage such work flows by:

1. facilitating specification of routing and distribution of data, information, and knowledge;
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.

While the AKM in its role as a static business process engine can easily support areas one and two in the above list, AKM agents may also 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 [9]. 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 [10] 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 [11]. 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 [12]. While this is not so much a contribution to process control in itself, it does support other agents in the AKMS 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.

In addition to the above, the AKM supports management of work flows composed of tasks performed by multiple application servers of diverse processing type. For example, a collaborative planning work flow application involving a planning business process engine and multiple database servers can be integrated by an AKM. Another example is an integrated database marketing workflow involving ETML, Operational Data Store, DSS Database, Data Mining, Business Process Engine, and Web Server components.

- Transactional Multithreading

Transactional multi-threading is the ability to manage each thread within a process as a separate transaction. Each thread can represent an instance of an active object.

Because they support transactional multi-threading, AKMs provide for multiple objects, belonging to different classes, to reside in the same process. This form of multitasking allows for concurrent execution of disparate business rules associated with different objects. It provides the AKM with parallelism useful in work flow management as well as in object and component DI.

- Business Rule and Metadata Management and Processing

Business Rule and Metadata Management and Processing are both derivative services of Object and Component Management and Synchronization. Business Rules are encapsulated in objects and components as methods, while metadata is encapsulated as attributes. So part of what we mean when we refer to object and component state management and synchronization is management and processing of business rules and metadata.

In-Memory Active Object Model/Persistent Object Store

The AKM provides an Active Object Model. It is distributed. Much of it is shared across physical platforms. And it can be either persistent or resident in-memory

- Event-Driven Behavior

Object methods in the Active Object Model are triggered by (1) events, (2) agents, and /or (3) programmed periodic activation. Events include user inputs, changes in object attribute values, changes in attributes themselves, or changes in methods themselves.

Events can trigger agent behavior, which then follows an autonomous course in implementing adjustments. Event-driven behavior is implemented in the AKMS through sequences of rules having antecedents and consequents.

- AKMS-wide Model With Shared Representation

Many of the objects in the AKM are shared across distributed physical platforms -- either data stores, or application servers. In fact, the AKM may be viewed as a special distributed application server or business process engine that maintains state, shares a set of reflexive objects across physical platforms, and manages and integrates multiple processes changing these shared objects. It is this sharing of objects and components across platforms that creates a common view of the AKM and its metadata. Figure Four illustrates the role of Shared Objects in the AKM and in DI.

- Declarative Business Rules

Both declarative and procedural business rule networks are supported as methods in classes and objects of the AKM model. Declarative Rule networks are those whose rules fire in parallel to determine an outcome. Procedural Rule networks are those whose rules fire in sequence. Figure Five Illustrates declarative and procedural rule networks.

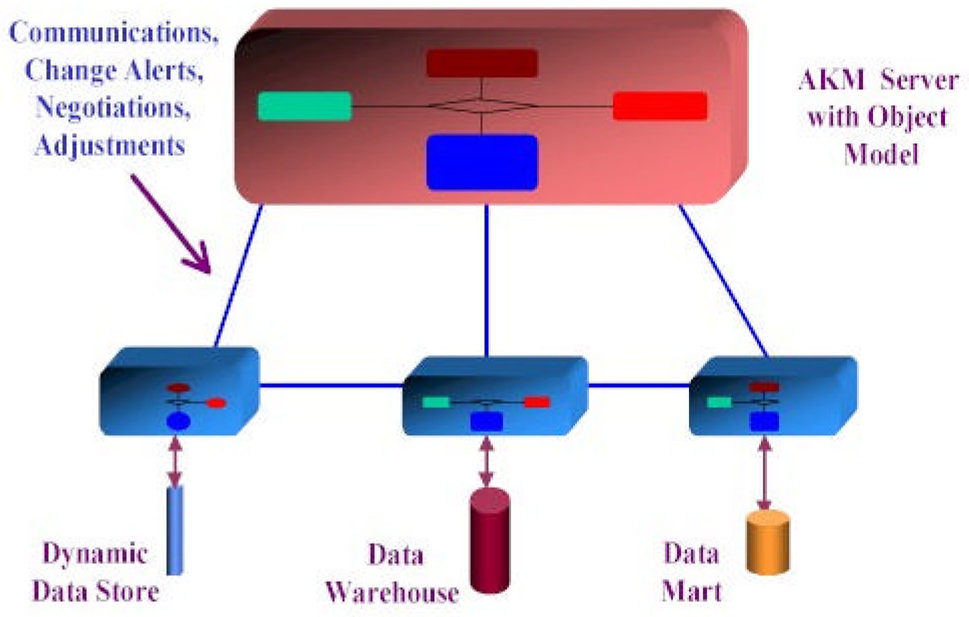
Communications, Change Alerts, Negotiations, Adjustments
AKM Server with Object Model
Dynamic Data Store
Data Warehouse
Data Mart

Figure Four -- A Distributed AKM, Shared Objects and Dynamic Integration

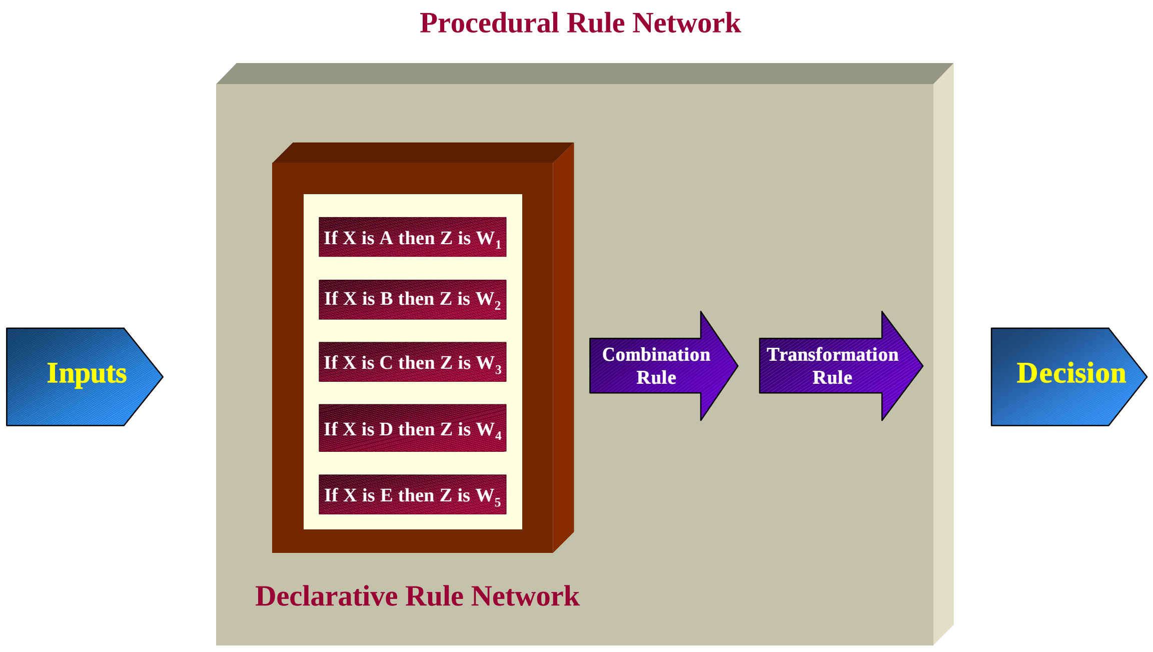

Figure Five -- Declarative and Procedural Rule Networks

Event-driven behavior in the AKM is frequently determined by sequences of declarative rules or rule networks constituting procedural rule networks. Agent-driven behavior is triggered by events but then is determined by the agent's autonomous program.

- Partial Instantiation of Objects

The ability to perform partial instantiation of objects is particularly important to the AKM in allowing it to develop rapid query performance. In partial instantiation only those attributes called for in a query, and only those records specified are brought into the in-memory object model. In this way, the data entering the AKM from data stores in the AKMS can be "chunked," and the amount of data that the AKM must handle can be minimized.

As a result, it is much more likely that the difficult processing involved in any query can be done in the AKM's "virtual database" in-memory. Figure Six illustrates partial instantiation of objects by an AKM.

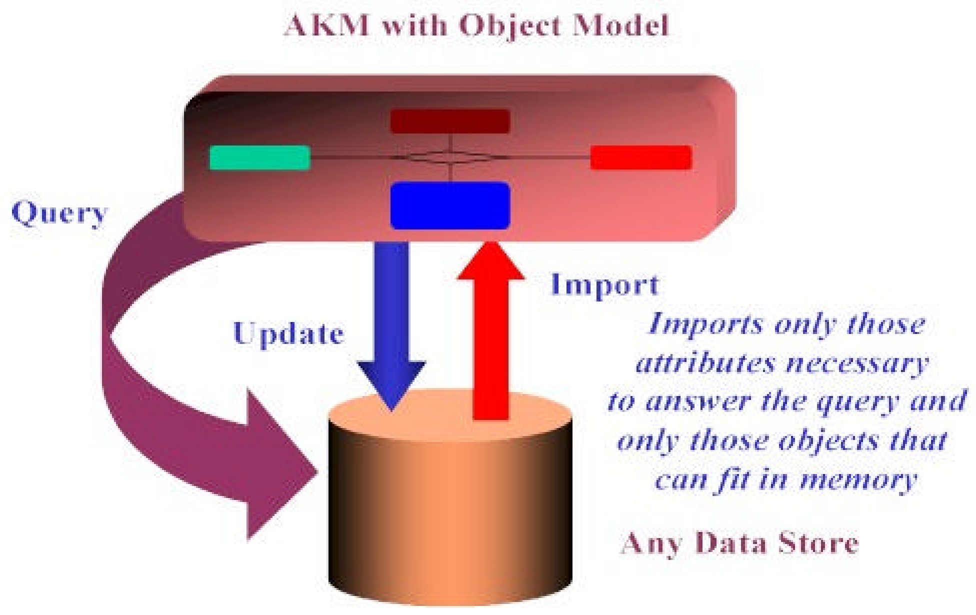
AKM with Object Model
Query
Update
Import
Imports only those attributes necessary to answer the query and only those objects that can fit in memory
Any Data Store

Figure Six -- The AKM and Partial Instantiation

- The Persistent Object Store

The AKM uses either a relational database or an OODBMS to store the Active Object Model in persistent form. In either case the Active Logical Object Model must be mapped to the physical data model of the database. The mapping is straightforward in case of an OODBMS, because the structure of the active object model matches the structure of the database. There is no "impedance mismatch," because there is no need to unwrap the logical objects and map their attributes onto physical table columns. This is the case with an RDBMS. And if one is used for persistent storage of the object model a performance penalty is paid.

- Reflexive Objects

AKMs use reflexive objects. Objects are reflexive if they are aware of their present state and any change of state. In this way they are like human agents in Natural Knowledge Management and other business processes. When combined with event-driven behavior reflexive objects provide the foundation for automatic propagation of events and changes in state among themselves.

- Software Agents

A Software Agent (SA) is an object that acts on behalf of another 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). [13] An intelligent software agent is an SA that: has an in-memory knowledge base including cognitions, evaluations, goals, and perhaps even affects [14]; 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 [15] 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.

The "source computer" of a mobile SA is its home agency. [16] The agency consists of a computing environment, an agent scripting capability, and a database. The AKM is a home agency for mobile agents. And may, in itself be viewed as a static agent as well as a business process engine (see below).

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.

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. [17] This environment is the AKM. And, in a larger sense, the AKMS.

Application Servers

The development of multi-tier distributed processing systems was characterized by the appearance of application servers. Application servers provide services to other components in a distributed processing system by executing business logic and data logic on data accessed from database servers.

The class of application servers is sub-divided by Rymer’s [18] distinction between "stateless" and in-memory server environments. Application Servers with Active in-memory Object Models he calls Business Process Engines (BPEs), a name similar to Vaskevitch's [19] Business Process Automation Engines.

Stateless Application Servers

According to Rymer: "Business state is the information that describes the momentary status of the organization. To create business state, most applications acquire data from a database and then load it into memory for manipulations by the user." [20] This is the "stateless" approach because, in it, a back-end database, rather than internal memory, manages state.

Among stateless application servers Rymer distinguishes:

- Web Information Servers (they provide access to databases from web browsers)
- Component Servers (they "provide data access and interaction frameworks for software components"); and
- Transaction Processing Monitors (they coordinate transactions within a distributed system).

Business Process Engines: Application Servers that Maintain State

"Business Process Engines manage the most important business state both in a fast in-memory environment and in close coordination with back-end databases." [21] Because of their in-memory maintenance of state, BPEs process many user requests without help from a database. In addition, they specialize in complex business rule processing, because their ability to maintain state is a special advantage in performing such processing.

KM software applications such as KDD/data mining servers, publication and delivery servers, the AKM itself, and many other server types are all BPEs. The job of the AKMS is to integrate the burgeoning list of BPEs into an enterprise wide system.

Therefore, an important aspect of specifying the AKMS is specifying the current universe of application servers and projecting the appearance of new types. Here are some criteria for defining types of Business Process Engines:

- whether they are distributed across physical components or not;
- whether a BPE application server deals with a single or multiple business processes; and
- the business process the BPE supports.

Distributed BPEs can be a powerful tool for upgrading performance in AKMSs, as well as for integrating their various components. An AKM is just a BPE that is both distributed and encompasses all of an AKMS's processes. A multi-process BPE can fall short of being an AKM, and instead can be restricted to a cluster of related processes. So, there are at least three types suggested by this criterion: a single process BPE, a BPE cluster, and an AKM.

How well a multi-process BPE performs will be correlated to the extent of its distribution, and to the complexity of the process it must support. But holding complexity constant, single process, non- distributed BPEs will generally perform better than multi-process non-distributed BPEs. So, multi-process BPEs will generally be distributed BPEs.

The third criterion for classifying BPEs is the business process supported. Here is an incomplete classification of BPE application servers based on knowledge, KM and Data Warehousing sub-processes. Collaborative Planning; Extraction, Transformation, and Loading (ETL); Knowledge Discovery in Databases (KDD); Knowledge base/object/component model maintenance and change management (The AKM); Knowledge Publication and Delivery (KPD); Computer-

Based Training (CBT); Report Production and Delivery (RPD); ROLAP; Operational Data Store (ODS) Application Server; Forecasting/ Simulation Server; ERP servers, Financial Risk Management, Telecommunications Service Provisioning, Transportation Scheduling, Stock Trading Servers, Work Flow servers.

Object/Data Stores

There are few, if any, limits on the types of object/data stores in the AKMS. These data stores incorporate objects, components, or their attributes in a non-volatile persistent form.

Legacy data, flat files, Relational Data Bases, Object Relational Data Bases, OODBMSs, multidimensional data stores, and vertical technology databases all fit within the AKMS.

In addition, the AKMS must also integrate Image, Text, Report, Video, Audio, and File Document Types. That is, it is the job of the AKMS to develop and maintain connectivity to various data stores, and not simply DBMSs.

Object Request Brokers

ORBs provide an intermediate layer between clients and servers in a distributed network. The ORB receives requests from clients and selects servers to satisfy the requests. The ORB can activate appropriate servers. The ORB can translate data between clients and servers. Generally, ORB servers are stateless and therefore are not BPEs (though this is not a necessary consequence of ORB specifications).

The AKM must support CORBA and DCOM ORBs to fulfill its integrative function. That is, it must be able to act as both CORBA and DCOM Servers and Clients. In this way, the AKM, with its greater integrative functionality, is built "on top of" an ORB standard.

The Unified Knowledge Language (UKL)

The KMCI is currently developing a standard on the Unified Knowledge Language (UKL). This standard will specify a contextually rich language that can represent and transmit knowledge from one software program or device to another. The language will consist of syntax, rules, and format. The specification will contain a message structure for the transmission of knowledge via the Artificial Knowledge Manager. So, as time goes on the AKMSC will need to coordinate with the UKL Committee in refining our standard.

AKMS Use Cases

When an AKMS is viewed functionally as an application, it performs a set of use cases supporting various tasks within the main activities of the knowledge and KM processes of an enterprise. The Kn processes of an enterprise are:

- Knowledge Production;
- Knowledge Acquisition;
- Knowledge Transmission.

The KM processes are [22]:

- Representing KM
- Leading KM
- KM Knowledge Production
- KM Knowledge Acquisition
- KM Knowledge Transmission
- Changing Knowledge Process Rules
- Handling Crises in Knowledge Processes
- Allocating KM Resources and mandating implementation for Various Knowledge and KM Process activities
- Negotiating KM with business process representatives

Here are three side-by-side lists (See Table One): a list of knowledge processes, one of associated Kn and KM activities, and a list of AKMS use cases that might be implemented in a comprehensive KM enterprise wide application. These also illustrate the point of partial support of the activities and the processes by the AKMS use cases. This list is intended as a start toward specifying AKMS use cases. The AKMSC can work over the list, and develop use case descriptions and an appropriate use case model. These products can provide a standardized view of the AKMS that we can then match against its component model to validate it.

Table One Kn and KM Processes and AKMS Use Cases

Knowledge & KM Processes	Activities Within Processes	AKMS Use Cases
Knowledge Production	Searching (within the enterprise for data, information, or knowledge)	• Perform cataloging, and tracking of previously acquired enterprise data, information, and knowledge bases related to business processes
	Receiving (transmitted data, information, or knowledge)	• Receiving transmitted data, information, or knowledge through e-mail, automated alerts, and data, information, and knowledge base updates
	Storing (and loading data, information and knowledge)	• Storing the outcomes of searching, receiving, and other knowledge production activities into a data, information or knowledge store accessible through electronic queries
	Retrieving (data, information or knowledge)	• Retrieve through computer-based querying data, information, and knowledge of the following types: • Planning • Descriptive • Cause-effect • Predictive and time-series forecasting • Assessment
	Formulating new knowledge	• Prepare data, information, and

	claims	knowledge for analytical modeling • Perform Modeling including revising, reformulating, and formulating models
	Testing knowledge models and claims	• Testing competing knowledge models and claims using appropriate analytical techniques, data, and validation criteria
	Concluding (about knowledge models and claims)	• Assessing test results and comparing (rating) competing knowledge models and claims
	Using Previously available Knowledge	
Knowledge Acquisition	Searching (for data, information, or claimed knowledge external to the enterprise)	• Perform cataloging, and tracking of external data, information, and knowledge bases related to enterprise business processes
	Gathering (externally located data, information or claimed knowledge)	• Order data, information, or external claimed knowledge and have it shipped from external source
	Purchasing (data, information, or claimed knowledge)	• Purchase data, information, or external claimed knowledge
	Filtering (including cleaning, transforming and staging data, information, or knowledge claims)	• Extract, Reformat, Scrub, Transform, Stage, and Load, data, information, and knowledge claims acquired from external sources

	Testing knowledge models and claims from external sources	• Testing competing knowledge models and claims using appropriate analytical techniques, data, and validation criteria
	Concluding (about knowledge models and claims from external sources)	• Assessing test results and comparing (rating) competing knowledge models and claims
	Storing (and loading data, information, and knowledge)	• Storing the outcomes of Filtering, Concluding, and other knowledge production activities into a data, information or knowledge store accessible through electronic queries
	Using Previously available Knowledge	
Knowledge Transmission	"Pushing," data, information and knowledge	• Publish, disseminate data, information, and Knowledge using the enterprise intranet • Update all data, information, and knowledge stores to maintain consistency with changes introduced into the AKMS
	Sharing knowledge (by making it available)	• Load data, information, or knowledge and updates into enterprise stores and provide access to enterprise query and reporting tools
	Searching/retrieving data, information, and knowledge using an electronic network	• Search/retrieve from enterprise stores through computer-based querying, data, information, and knowledge of the following types:

		• Planning • Descriptive • Cause-effect • Predictive and time-series forecasting • Assessment
	Searching/retrieving data, information, and knowledge using personal networks	• Use e-mail to request assistance from personal networks
	Face-to-face knowledge transmission within small groups	
	Face-to-face knowledge transmission in formal training and education	• Present knowledge using computer-aided displays
	Storing (and loading data, information, and knowledge)	• Storing the outcomes of knowledge transmission activities into a data, information or knowledge store accessible through electronic queries
	Using Previously available Knowledge	
Representing	Signing Contracts	

(at ceremonies)		
	Attending Public Functions for KM	
	Meeting and relating to dignitaries	
Leadership	Hiring KM Staff	• Identify Knowledge Management responsibilities based on some segmentation or decomposition of the KM Process • Retrieve available qualification information on knowledge management candidates for appointment • Evaluate available candidates according to rules relating qualifications to predicted performance • Communicate Appointments to Knowledge Management constituency
	Providing for KM Staff Training	• Plan or select training program(s) • Purchase or create training vehicles and materials (seminars, CBT products, manuals,etc.)
	Motivating KM Staff	• Plan and Schedule motivational events
	Monitoring KM Staff	• Querying and Reporting using data, information, and knowledge about:

		• KM staff plans • KM staff performance description • KM staff performance cause/effect analysis • KM staff performance prediction and forecasting
	Evaluating KM Staff	• Querying and Reporting using data, information, and knowledge about assessing KM staff performance in terms of costs and benefits
	Building relationships with individuals and organizations external to the enterprise	• Communicate with external individuals through e-mail and online conferencing technology
KM Knowledge Production	Activities are the same as those specified for Knowledge Production	• AKMS use cases are analogous
KM Knowledge Acquisition	Activities are the same as those specified for Knowledge Acquisition	• AKMS use cases are analogous
KM Knowledge Transmission	Activities are the same as those specified for Knowledge Transmission	• AKMS use cases are analogous
Changing Knowledge Process Rules	Deciding to change Knowledge Process Rules	• Search/retrieve from enterprise stores through computer-based querying, data, information, and knowledge of the following types about knowledge process rules:

		• Planning • Descriptive • Cause-effect • Predictive and time-series forecasting • Assessment
	Directing use of KM Knowledge Transmission to transmit new rules and mandate for using them	• Communicate directives through e-mail
Crisis Handling	Various activities associated with other processes implemented in a "time box" mode	• Uses AKMS use cases associated with those activities
	Forecasting developing crises or crisis potential	• Search/retrieve from enterprise stores through computer-based querying and reporting, data, information, and knowledge of the following types about crisis potential: • Cause-effect, and • Predictive and time-series forecasting
	Monitoring developing crises	• Search/retrieve from enterprise stores through computer-based querying, data, information, and knowledge of the following types about crisis potential: • Descriptive • Cause-effect

	Evaluating developing crises	• Search/retrieve from enterprise stores through computer-based querying, data, information, and knowledge about crisis potential of the following types: • Descriptive • Cause-effect • Predictive and time-series forecasting • Assessment
	Contingency Planning for crisis	• Search/retrieve from enterprise stores through computer-based querying, data, information, and knowledge of the following types about crisis potential: • Planning • Descriptive • Cause-effect • Predictive and time-series forecasting • Assessment
	Evaluating crisis contingency plans against crisis management performance	• Search/retrieve from enterprise stores through computer-based querying, data, information, and knowledge of the following types about crisis potential: • Planning • Descriptive • Cause-effect • Predictive and time-series forecasting • Assessment

Allocating KM Resources and mandating implementa-tion for:	KM infrastructure	• Specify (either alone or using a work group) and compare alternative KM infrastructure options in terms of anticipated costs and benefits • Communicate directives through e-mail
	Training	• Specify (either alone or using a work group) and compare alternative KM training options in terms of anticipated costs and benefits • Communicate directives through e-mail
	Professional Conferences	• Specify (either alone or using a work group) and compare alternative KM professional conference options in terms of anticipated costs and benefits • Communicate directives through e-mail
	Compensation for KM staff	• Specify (either alone or using a work group) and compare alternative KM compensation options in terms of anticipated costs and benefits • Communicate directives through e-mail
	Funds for new KM programs	• Specify (either alone or using a work group) and compare alternative KM budget options in terms of anticipated costs and benefits • Communicate directives through e-mail
Negotiating with business		• Specify (either alone or using a work group) and compare alternative KM

process representa-tives about:	Levels of effort for KM	level of effort options in terms of anticipated costs and benefits • Communicate options, proposals, and responses through e-mail and online conferencing and collaboration application
	The scope and content of KM programs	• Specify (either alone or using a work group) and compare alternative KM scope and content options in terms of anticipated costs and benefits • Communicate options, proposals, and responses through e-mail and online conferencing and collaboration application
	KM ROI targets	• Specify (either alone or using a work group) and compare alternative KM ROI targeting options in terms of anticipated costs and benefits • Communicate options, proposals, and responses through e-mail and online conferencing and collaboration application
	KM infrastructure support for business processes	• Specify (either alone or using a work group) and compare alternative KM infrastructure support options in terms of anticipated costs and benefits • Communicate options, proposals, and responses through e-mail and online conferencing and collaboration application
	KM staff support for business processes	• Specify (either alone or using a work group) and compare alternative KM staff support options in terms of anticipated costs and benefits

		• Communicate options, proposals, and responses through e-mail and online conferencing and collaboration application

<u>Conclusion: AKMS Committee Sessions and Program</u>

I intend that this working paper be used a starting point for the AKMSC. That means the committee could junk the framework of the paper in favor of some alternative, modify it substantially, or perhaps use it as a foundation, and flesh out a more complete and precise picture. We can start the process at the standards committee facilitated AKMSC sessions on the AKM standard and Artificial Knowledge Base Management Systems on January 29, 1999.

Figures Two and Three above, suggest a basic "straw man" program for specifying the AKMS and the AKM Standard. It is as follows:

1. Specify AKMS Use Case Model and Relate to NKMS Processes and Activities
2. Specify the Artificial Knowledge Manager (AKM) Logical Component
3. Specify Types of Client Application Components.
4. Specify Types of Application Servers
5. Specify Communication Buses including Object Request Brokers (ORBs)
6. Specify Types of Data Stores
7. Specify AKMS Architectural Model
8. Specify AKMS Model
9. Specify Artificial Knowledge Manager Standard
10. Specify Knowledge Warehouse Standard

Clearly, there's an appreciable amount of work associated with these specification tasks. We can begin work at the January 29 meeting, first considering whether to retain, modify, or reinvent the straw man program. Then we can divide into subcommittees according to our interests and specialization in order to do the work over the longer term.

References

[1] Compare Edward Swanstrom, "What is Knowledge Management?" Discussion Rough Draft, 1998.

[2] John H. Holland, Hidden Order (Reading, Mass.: Addison-Wesley, 1995)

[3] John H. Holland, Emergence (Reading, Mass.: Addison-Wesley, 1998)

[4] M. Mitchell Waldrop, Complexity (New York: Simon and Schuster, 1992)

[5] See Joseph M. Firestone, "Distributed Knowledge Management Systems and Enterprise Knowledge Management Modeling," at http://www.dkms.com/White_Papers.htm.

[6] The AKMS concept developed here is largely based on the DKMS concept I introduced in "Object-Oriented Data Warehousing," available at http://www.dkms.com/White_Papers.htm. Other papers developing various aspects of the DKMS are: Joseph M. Firestone, "Distributed Knowledge Management Systems: The Next Wave in DSS," Joseph M. Firestone,"Architectural Evolution in Data Warehousing," Joseph M. Firestone, "Knowledge Management Metrics Development: A Technical Approach," Joseph M. Firestone, "DKMS Brief No. Four: Business Process Engines in Distributed Knowledge Management Systems," all are available at http://www.dkms.com/White_Papers.htm, as are additional papers about the DKMS.

[7] The ideas for the AKM owe much to the following White Papers. Template Software, "Integration Solutions for the Real-Time Enterprise: EIT - Enterprise Integration Template," Dulles, VA, White Paper May 8, 1998. See also http://www.template.com. Persistence Software, "The PowerTier Server: A Technical Overview" at http://www.persistence.com/products/tech_overview.html, and John Rymer, "Business Process Engines, A New Category of Server Software, Will Burst the Barriers in Distributed Application Performance Engines," Emeryville, CA, Upstream Consulting White Paper, April 7, 1998 at http://www.persistence.com/products/wp_rymer.html.

[8] Reflexive objects are used in Template Software Enterprise Integration Template and DAMAN Consulting's InfoManager products. 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. DAMAN is at http://www.damanconsulting.com.

[9] Both the Enterprise Integration Template and InfoManager, provide a distributed object model as well as process control and connectivity services useful in developing a distributed AKM. See Ibid.

[10] Information Retrieval Agents similar to what I've called knowledge retrieval agents are conceptualized in Jeffrey M. Bradshaw (ed.), Software Agents (Cambridge, MA: AAAI Press/M.I.T. Press, 1998), Pp.11-12. I've relied on this development in my treatment.

[11] 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.

[12] See Bradshaw (ed.), Software Agents . . . P. 31.

[13] 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:

[14] See Rosalind W. Picard, Affective Computing (Cambridge, Mass.: MIT Press, 1997).

[15] 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

[16] Ibid.

[17] Bradshaw (ed.). "Software Agents . . ." Pp. 26-27

[18] John Rymer, "Business Process Engines, A New Category of Server Software, Will Burst the Barriers in Distributed Application Performance Engines," Emeryville, CA, Upstream Consulting White Paper, April 7, 1998, P. 1, at http://www.persistence.com/products/wp_rymer.htm.

[19] David Vaskevitch, Client/Server Strategies (San Mateo, CA: IDG Books, 1993) Ch. 8.

[20] Rymer . . ., P. 1.

[21] Ibid. P. 9.

[22] I've discussed both Kn and KM processes in " Distributed Knowledge Management Systems and Enterprise Knowledge Management Modeling," available at http://www.dkms.com/EKMDKMS.html.

Biography

Joseph M. Firestone, Ph.D. is an Information Technology consultant working in the areas of Decision Support (especially Enterprise Knowledge Portals, Data Warehouses/Data Marts, and Data Mining), and Knowledge Management. He is consulting in the areas of developing Enterprise Information/Knowledge Portal Products, and is the author of "Approaching Enterprise Information Portals," a comprehensive, full-length industry report on this rapidly emerging field. In addition, he formulated and is promoting the concept of Distributed Knowledge Management Systems (DKMS) as an organizing framework for software applications supporting Natural Knowledge Management Systems. Dr. Firestone is Chief Scientist of Executive Information Systems, Inc. (EIS), and one of the founding members of the Knowledge Management Consortium, International. A sampling of his writings may be found at the EIS web site at http://www.dkms.com, a site Dr. Firestone developed. The dkms.com web site is one of the more popular sites in data warehousing and knowledge management, and has now attained a run rate of more than 70,000 visits per year.