Intelligent Agents

Seth Grimes, grimes@access.digex.net

Database Programming and Design, April 98

To gauge how “hot” a technology is, try pinning down a definition that covers all the players and products that claim to implement it. For a cutting-edge technology like intelligent agents, you’ll come up with a grab bag: Beyond a germ (or two) of commonality among the definitions where all or most of the candidates agree, many products and research efforts take fundamental agent concepts and properties in widely divergent directions.

“The term *agent* is used to represent two orthogonal concepts. The first is the agent's ability for autonomous execution. The second is the agent's ability to perform domain oriented reasoning.” This definition, from Sankar Virdhagriswaran of agent technology company Crystaliz Inc., is a good starting point but can be expanded in certain areas. First, an agent executes; it acquires input and produces output. Agent processing is domain-oriented: An agent “knows” about certain concepts, data structures, rules, and interfaces but is not necessarily capable of interpreting information outside its field. An agent also “reasons” by encapsulating rules that allow it to transform conditions into decisions. And it operates autonomously by virtue of being persistent and capable of operating in a changing environment.

So far, our definition of an intelligent agent applies equally well to travel agents, to many industrial robots, and to software agents. But it’s software, of course, that interests us here rather than robot hardware or human “wetware”--database-related software in particular. This article will introduce software-agent technology and concentrate specifically on ways that agents add value to database-related processes such as knowledge discovery and other forms of data analysis, profiling and search, and reporting. In the database realm, much or even most of the relevant work is being done not with mainstream relational database management systems but rather with specialized DBMSs that manage analytical data and unstructured information like text, images, and multimedia Web content. Agents are also able to encapsulate pattern-recognition algorithms and rules-based processing systems that make them a potentially ideal tool for

automation of data-centric tasks. No article on agents would be complete, however, without discussing some of the most interesting agent technology, technology that will surely have a role in the database world: mobile agents and *bots*, agents that interact with other agents.

Concepts and Implementations

Software agents are computer programs, different from nonagent programs in their ability to run autonomously, sensing and acting on changing environmental conditions. Because they run autonomously, they must be self-contained, with data structures, methods, and interfaces necessary to interact with the operating environment (also known as the "agent subsystem") that is layered on top of the operating system. The structure of an agent program is clearly similar to that of a software object, and in practice agents are typically implemented as C++ or Java classes (even if they are coded in higher-level, rules-based languages) or Tcl, Telescript, or Python scripts. What differentiates agents from other "strongly encapsulated" software objects is their behavior rather than the programming languages in which they are written. This behavior is determined and regulated by three factors: interfaces, operating environment, and encapsulation of rules, procedures, and event (condition) handling.

Agent interfaces and communications. Agents are autonomous and self-contained, but interaction with dynamic environments--which occurs through published APIs and message-based protocols--is their *raison d'être*. Agents that communicate with standard computing services such as operating systems, Web servers, and DBMS servers will use the same protocols as nonagent programs. In the case of multiagent or agent-manager systems, protocols may be proprietary or based on one of the (emerging) standard agent communications languages. Agent communication protocols should not be confused with transport mechanisms such as CORBA, Java Remote Method Invocation (RMI) or JavaBeans, Microsoft COM, or TCP/IP sockets.

The de facto standard for interagent communications is Knowledge Query and Manipulation Language (KQML). The Foundation for Intelligent Physical Agents (FIPA), a nonprofit association for agent-based applications, services, and equipment,

proposes an alternative, Agent Communication Language (ACL). Here's a schematic example that shows the structure of a message:

[[Can you make sure that the second "content" is indented if it's on a separate line from the first? It's actually part of the first.]]

(**request-whenever**

_____**:sender** agent99

_____**:receiver** agent87

_____**:content** ((inform :sender agent87 :receiver chief

:content (location))(= (time) 00))

_____**:in-reply-to** agent87-3

_____**:reply-with** code2

_____**:language** accadian

_____**:protocol** control

_____**:ontology** chaos

)

The message has a *communicative act type* (request-whenever) that is also known as a *performative*, a *content expression*, and *message parameters* that describe and form an envelop for the content. The ontology is domain-specific; the language could have been Java, Knowledge Interchange Format (KIF), Perl or Python, SQL, Tcl, or just about anything that the receiver can understand. The receiver parses and processes the message, communicating with the back-end database and other servers as necessary.

Back-end communications are the same for agents as for nonagent client or server software, typically involving proprietary data formats and interchange protocols tailored by software designers to their applications. But because the purpose of agents is often to find, classify, and index information in diverse formats from heterogeneous sources, agents that work with data sources have special interpretive needs. Popular formats are always open in the sense that programs other than the word processors, photo editors, and spreadsheets that created them can read them. Formats designed simply as data stores usually include little descriptive information; programs that use them depend on MIME

types (such as image/jpeg or text/html) and metadata managed by the operating system--name, creator, creation date, and so on--to provide context. This type of envelope information does not describe content elements and is insufficient for agent processing of diverse networked documents.

Extensible Markup Language (XML) is a recent innovation that may lead to easier content processing by agents. With its reintroduction of the content-element specification found in Standard Generalized Markup Language (SGML) to SGML's sickly offspring, Hypertext Markup Language (HTML), XML provides knowledge-domain-related hooks for agents to interpret Web content directly rather than through potentially less accurate content analysis. With its steady progress through World Wide Web Consortium processes for establishing Web standards (see http://www.w3.org/XML/), XML is poised to become the leading Internet data format for structured document interchange and thus will be supported by agents that need to access Internet data sources.

Agent subsystems. An agent subsystem provides scheduling, memory management, I/O, network and interprocess communications, and other services to agents. Examples could range from an operating system where agents run as daemon processes, to a Java virtual machine, to a DBMS or application server process or thread. FIPA uses the term Agent Platform (AP) for the agent operating environment, providing a message-routing Agent Communication Channel (ACC), a Directory Facilitator (DF) for registration and location of services, and an Agent Management System (AMS) that "manages the creation, deletion, suspension, resumption, authentication, and migration of agents on the agent platform and provides a 'white pages' directory service for all agents resident on an agent platform."

Multiagent systems of communicating, cooperating agents are a special case. Agents in such a system send messages to one another directly or through agent managers. Agents may also be mobile rather than stationary. A mobile agent requires mechanisms for locating other suitable and desirable environments, negotiating their use, propagating its code and data including its state to them, and possibly shutting down its earlier instance. That's cool: the type of software that has most successfully implemented this "wherever I hang my hat is my home" computing mode so far consists of viruses and

the like. Java applets, JavaScript programs, and ActiveX controls, of course, run in this mode but few of them merit the “agent” label.

Some of the sexiest agent work is being done with multiagent and mobile-agent systems for stuff like Internet shopping and auctions. Frankly, I’m not ready to give a software agent my credit card: If my human administrative assistant couldn’t understand that booking a flight for me from Washington, D.C. to Miami via Chicago is not an attractive route, especially when it doesn’t save any money, then I’m not going to trust that a *bot* can.

Infrastructures to support mobile agents are being developed for distributed object computing architectures. For instance, leading vendors and the Open Group are contributing to a new Mobile Agent Facility (MAF) that will be added to the Object Management Group’s CORBA standard. On the Java front, the Java Management API specifies RMI mechanisms for Java agent management.

Domain-based reasoning. A reasoning system has three principle components: symbolically represented facts that describe the environment and the system state, a knowledge base of domain-specific information and rules, and an execution control facility for rules. The rules themselves are parameterized statements of the form “if (condition) then (action).”

AI researchers have studied machine reasoning for years and many different languages have been used or developed to process rules. Agent operating requirements differ, however, from those of conventional inference engines. While agents must often continue to support complex sets of overlapping, conflicting rules, they have special security and communications needs. Interpreted scripting languages and Java now appear to lead other options for coding rules for agent encapsulation.

Tools that provide specialized graphical rule-building and maintenance interfaces are an attractive alternative to generic coding products. ILOG Rules (http://www.ilog.com) is one example. In ILOG’s definition, rules consist of a name, priority, set of conditions represented by filters that are applied to the objects the rule acts on, and set of actions applied to the filtered objects. ILOG rules are coded in a high-level, object-oriented language that is compiled into C++ or Java classes; they may also

be interpreted at runtime. An ILOG Rules agent is simply an instance of a rule class supplemented by a "context" rule set describing agent behavior.

Working Agents

"Agent" is a hot buzzword applied to diverse technologies and products. I'll survey representative examples from the database world chosen to show how researchers and vendors have created practical applications embodying agent concepts: applications that go beyond merely automating human-machine interactions. Database processes of interest include (but are not limited to) data analysis and reporting, database and systems management, and indexing, profiling, and search.

Data analysis and reporting. In "The New Face of Data Access" (*Database Programming & Design*, February 1998), I wrote that query, reporting, and analysis vendors are only starting to incorporate agent technology into their products. Agent-type functions appeared first in enterprise query and reporting systems; support for stored query and report templates, which are analogous to interest profiles, fired on a calendar or event basis, is now common.

Some data-access agents will monitor data events--for example, watching a set of indicators (also known as metrics or derived variables) and acting when the take on certain values. This type of condition handling is essential in the case of systems with volatile data: Because updates are frequent, an agent that acts on every data-loading event wouldn't be suitable.

Tools choices are usually made on reporting and analytical capabilities, interfaces to databases and desktop software, and the ability to handle heavy workloads and diverse types of use. Support for distributed-object and Internet computing has recently joined these decision factors, and a complete strategy will include agent support. I'll single out only one vendor, Andyne Computing, whose approach to the Internet and agents is comprehensive and modern and whose tools are flexible and offer a good combination of analytical and reporting capabilities.

Database and systems management. After working dozens of times a week through the minimally formatted results of manually typed commands that check the status of log files, disk space, and database server processes, database and systems

administrators realize that to remain sane they have to find an easier way to monitor their software. A first step is to create command aliases as well as commands that extract sought information from output. Monitoring with shorthand commands is still labor-intensive, however, and there are two paths to automating the task further.

One approach is to move to full-blown scripts run periodically with cron or another scheduling mechanism. These scripts can include logic that sends an email message or page when conditions like free disk space falling below a threshold value are met. Another option is to install one of the many DBMS and system monitoring utilities available on the market that allow you to specify monitoring events and alarm thresholds and present status information is graphical "dashboard" format; they also save you from having to decipher complex system catalogs.

Automated monitoring programs fit the agent definition: they run autonomously as daemon processes or via a scheduling program, they incorporate domain-based knowledge about DBMS and system data structures and domain-based reasoning capabilities that they apply to dynamic environments, and they process and act on the information they collect. (Most roll-your-own monitoring programs don't accept messages, of course.)

Unicenter TNG, an "enterprise management" tool from Computer Associates, claims to belong to "The Next Generation" in that it centralizes management of disparate systems, network, and database resources through a single, repository-centered framework. For supported DBMSs--CA-Datacom, CA-IDMS, DB2, Informix, OpenIngres, Oracle, and Sybase--it deploys distributed agents that monitor status, event, performance, space, and configuration information and alert the Unicenter TNG agent manager(s) about events according to user-defined action and escalation policies. The manager in turn takes corrective steps or sends a notification message. The Unicenter TNG Framework consists of an Enterprise Management GUI and Distributed Services including a repository, calendar and event management, reporting, and other infrastructure facilities that allow both client-server and Internet deployment. Unicenter TNG solutions such as the database agents are built from three application-programming interfaces, the CA-WorldView user interface API; the Enterprise Management API for

management functions, common services, and cross-application integration; and the Agent Factory API for construction of agents and managers.

Agents are also a key part of BMC Software's Cooperative Enterprise Management Solutions (CEMS), which is centered on its Patrol application and data management product. Patrol performs functions similar to those of Unicenter TNG and has a similar distributed agent computing architecture.

Web indexing, profiling, and search. Conceptually, the Web is a dynamic library of active, linked documents that is most often accessed with a browser. Search engines help users locate information and services by gathering content and descriptive information and creating indexes; an AltaVista, Excite, or Yahoo search is really, of course, a search of the AltaVista, Excite, or Yahoo Internet index database, and returned hits are often irrelevant or dead (out of date). Sites such as Firefly (http://www.firefly.com) and tools like Learn Sesame (http://www.opensesame.com) that base their indexes on "collaborative filtering" seek to improve relevance by capturing in their indexes intentional information such as user ratings and recommendations. They typically require users to create server-based profiles used to guide them in navigating Web sites and identifying content rather than just locating pages.

Alexa Internet combines the major advantages of both models but avoids their architectural shortcomings. Alexa is a browser companion, a desktop agent that maintains a non-HTTP connection to Alexa's Internet site in parallel with the browser's Web site connection. Alexa software monitors the browser and communicates navigation actions to Alexa's server; it receives Web site metainformation including traffic level, domain ownership, ratings, and links to similar sites. These links are based on recorded usage paths, on content link analysis--what sites link to this site and what sites does it link to?--and on content analysis performed with Aptex's Convectis engine (http://www.aptex..com).

There are many collaborative filtering and Web search tools as well as desktop agents. These desktop agents range from email filters, to word-processing helpers that automatically correct spelling, format text, and provide content-sensitive help as you type, to IBM's Web Browser Intelligence (WBI) personal navigation agent, but Alexa is the only instance I've found of a product that combines these functions. Alexa also

provides two unique features: access to the 8TB Internet archive (http://www.archive.org) created by Alexa founder Brewster Kahle and location-sensitive advertising. Users are given the ability to retrieve pages--including expired ones whose URLs bring up HTTP's "404 Not Found" message--from the Archive's tape jukebox system. The banners ads, intended of course to pay for and profit from Alexa, could potentially be targeted to users based on the pages they're visiting and on their Alexa profiles.

Alexa is working software--the client program is available by free download from www.alexa.com--but it will be some time before the Alexa system fulfills its potential as a navigation aid. At press time, Alexa provided navigation aid at the site rather than the page level. Site recommendations were often-as-not of little use presumably due to a small database of usage paths and to the fact that user ratings (currently just a like/dislike vote for an entire site) are simplistic, no personal profiling is available, and capacity for acquisition and indexing of new sites is apparently limited. These are component-related rather than architectural concerns: Alexa intends to move to page-level analysis, to expand user ratings, to use Open Profiling Standard information and allow users to enter demographic information and maintain personal profiles, and to use metainformation provided by XML tags; furthermore, and the usage database is constantly growing. Indexing capacity should also increase as Alexa gains revenue and additional funding.

Systems for Building Agents

I've sketched out the agent-building facilities provided for one commercial product, Unicenter TNG, although these facilities are limited to creation of agents for system and database management tasks operating within a proprietary framework and using proprietary APIs. There are quite a few other efforts underway to define agent frameworks, frameworks like CA's that are platform-independent but differ in permitting extension of APIs and base services. Now I'll introduce an example from one class, Java agent frameworks, and discuss how object/relational database management systems could serve as an agent environment.

Java agent frameworks. Java Agent Template, Lite (JATLite) is a typical Java agent framework: a set of Java classes and infrastructure programs for template-based

creation of network agents. JATLite is designed to work with KQML agent communications over TCP/IP networks like the Internet and features five increasingly specialized layers that can be used with or substituted for other software providing equivalent services. JATLite agents are written in Java and will run in any software with a JDK 1.1 Virtual Machine, whether browser or server, on any platform the software runs on. Java ensures security in a number of ways such as by allowing agents to communicate only with the host that served them; peer-to-peer, interagent communications and back-end requests are mediated by an Agent Message Router (AMR).

The Java-based Agent Framework for Multi-Agent Systems (JAFMAS) from the University of Cincinnati (http://www.ececs.uc.edu/~abaker/JAFMAS/) is a similar although less mature system in the same category as JATLite; both are oriented toward "speech-act based multiagent systems." IBM's Aglet framework for mobile agents is available free in an alpha version at http://www.trl.ibm.co.jp/aglets/index.html. General Magic incorporates another set of Java classes for mobile agents, Odyssey, in its forthcoming Serengeti "virtual assistant"; the company has abandoned its Telescript mobile-agent system having recognized the necessity to move to Java. Check out Web information on another Java mobile agent system, Toshiba's planning-enabled Plangent, at http://www2.toshiba.co.jp/plangent/, if only for amusing graphics that play on the system's name. There are many more Java-based multiagent and mobile-agent frameworks available or under development in academia and industry.

Object/relational and object databases: Unfilled promise. ORDBMSs add the ability to manage objects--structured data with encapsulated methods--to the relational systems that have dominated the commercial database market since the late 1980s. ORDBMS user-defined types (UDTs) and user-defined functions (UDFs) allow application logic to be moved into the database although ORDBMS vendors take divergent database-extension approaches. These extensions--packages of UDTs, UDFs, and data structures called by the now well-known names Cartridges (Oracle), DataBlades (Informix), and Extenders (IBM)--provide facilities for management of time-series, image, spatial data, formatted text, and Web assets.

There could be advantages in security, performance, and transaction- and data-management capabilities in hosting agents in an extensible object or object/relational database server when the agents act primarily on DBMS information. An Informix-Dynamic Server agent DataBlade tied to a DataBlade implementation of Excalibur's RetrievalWare text-management and profiling tools is slated for early 1998 availability. RetrievalWare and Excalibur's text-search blade offers extended options such as concept- and proximity-based search, fuzzy logic, thematic clustering, summarization, and creation of topic sets. The accompanying Real-time Profiling DataBlade will store user queries and allow them to be scheduled to execute when new data entered the database with user notification of hits dependent of reaching a relevance threshold. The intelligence of the two blades comes from their ability to represent knowledge through a concept-centered semantic network; profiles, however, are not modified based on user selections among hits or user ratings *à la* Alexa or the collaborative-filtering engines--that is, they don't learn. They also don't integrate with Excalibur's image search and multimedia Internet spidering tools. Excalibur says that learning capabilities and image-search profiling agents are being considered.

Another forthcoming Informix module, a [[Message]] DataBlade developed by Informix and Tibco, will offer push (publish and subscribe) capabilities [[across]] Informix applications [[and servers]]. It could also provide necessary event management facilities for agents running within the DBMS.

Note that object databases have extensibility equivalent to that of ORDBMSs and have been around longer, since the late 1980s. Object DBMSs have failed to take off, initially because they served as little more than persistent data stores for C++ and Smalltalk programs and lacked robust data-management features. They've overcome these deficiencies but still command limited market share and, more importantly, mind share. The biggest reason is that mainstream applications are likely to continue relying on tabular (relational) data structures; also, IBM, Informix, and Oracle are ahead in their ability to offer scalability features like parallelization and multiprocessing support. In general, applications are the key to acceptance of both object/relational and object databases, and it's not clear that vendors of either type of DBMS are going to deliver

applications, agent or other, that will compel users to migrate from RDBMSs and application-server-centric distributed object computing architectures any time soon.

I believe that users *will* migrate to ORDBMSs over the next five years, but it's not a sure bet that they will have a strong role as application servers or, similarly, agent-hosting environments. Vendors have taken only basic steps toward providing agent infrastructures, but given ORDBMS abilities to manage tightly and loosely integrated extensions invoked directly or via remote-procedure calls or CORBA and their management of both fielded and unstructured data, encapsulation of search and analytical methods, and user- and transaction-management capabilities, ORDBMS agent hosting is not far off.

The Forecast for Agents

Agent technology is hardly unique in that advances are being fueled and facilitated by Internet computing. Agents are poised to play an active mediating role between individuals and the Web's collection of disparate, distributed data source and services. In effect, agents will automate Web browsing. Many of the elements of an agent future are in place: search engines, desktop assistants, server managers, distributed object computing frameworks, virtual machines. While there are many competing agent approaches and no clear winners yet--there will never be a single type of agent that does everything--the need for software agents and their ability to deliver has already been demonstrated. Start getting used to agents rooting around in your data and on your desktop.

Suggested Resources

There's a wealth of material available on the Web if you're new to agents and want to know more. Go to Database Programming & Design Online (http://www.dbpd.com) for a few of the better ones.

Seth Grimes is a principal of Alta Plana Corp., a consultancy specializing in database design and software development for Internet and decision-support applications. Contact him through http://altaplana.com or at grimes@access.digex.net.