The Knowledge Engineering Review, Vol. 17:1, 81–85. © 2002, Cambridge University Press
DOI: 10.1017/S0269888902000322 Printed in the United Kingdom

Agents and ontologies for e-business

YANNIS LABROU

Fujitsu Laboratories of America, 8400 Baltimore Av., College Park, MD 20740, USA
e-mail: yannis@fla.fujitsu.com

Abstract

Academic work on agents and ontologies is often oblivious to the complexities and realities of enterprise computing. At the same time, the practitioners of enterprise computing, although they are adept at the building of robust, real-life enterprise applications, are unaware of the academic body of work and the opportunities of applying novel approaches of academic origin. Enterprise applications are very complex systems that are designed to support critical business operations. This article outlines the technical and business foundations of enterprise application software and briefly discusses viable opportunities for agents and ontology research.

1 Introduction and definitions

Although agents and ontologies are often part of the vocabulary of the practitioners of enterprise computing, the actual practice of the field relies heavily on tested and established methods. Databases are the cornerstone of enterprise applications and custom and costly integration across disparate enterprise systems is the usual means for providing additional business functionality. The current state of the art of enterprise computing creates a considerable barrier to the introduction of novel applications that attempt to make use of agents and ontologies.

Enterprises strive to take advantage of computing and networking advances in order to provide additional functionality and support increasingly complex business functions. The holy grail of the modern enterprise is e-business, which can be loosely defined as the discovery, procurement and sourcing of goods and services, over computer networks and within, but especially across, the boundaries of business units and enterprises. The challenge of conducting e-business across organisational boundaries is the space of opportunity for agents and ontologies. Agents, with the aid of ontologies, have been promoted as a paradigm for distributed computing in dynamic environments that require run-time autonomy and adaptivity. In theory, agents do seem appealing in the e-business context. Moving from theory to practice requires a rudimentary understanding of agents, ontologies, enterprise computing and the business functions it supports.

1.1 Ontologies

An ontology is a specification of a conceptualisation (Gruber, 1993), or, in layman terms, it is a description (specification) of a domain and of the "objects" that exist in that domain. There are three parts to an ontology: (1) conceptualisation – a hierarchy of concepts (with their attributes) and their relationships, which accounts for how one breaks down (conceptualises) the given domain; (2) vocabulary – the terms chosen for objects, relationships and properties (or attributes); and (3) axiomatisation – constraints (axioms) on the possible values for attributes and conditions that are applicable across attributes. Collectively, conceptualisation, vocabulary and axiomatisation define an ontology. There is no "correct" or "best" ontology for a particular domain or problem. Ontologies are always domain-specific and are designed for the realities of the particular domains and problems.

1.2 Agents

Agents began as an offshoot of artificial intelligence research. Despite the broad interest of researchers and practitioners from many subdisciplines of computer science for more than 10 years, the lack of convincing demonstrations of the advantages of an agent-based approach over "traditional" approaches, explains the hesitancy towards deploying them in real-world, critical applications. My view of agents is one of a distributed computing software paradigm. In this view, agents are not just about distributing computation among programs interacting in a networked environment. Agents are autonomous and adaptive and rather than being programmed for a particular problem they are programmed for a specific domain. Consequently, ontologies are indispensable to the development of successful agent systems because ontologies are the means for capturing the domain knowledge and context. According to one definition (Genesereth & Ketchpel, 1994), an agent is a piece of software that communicates with other agents through an agent communication language, such as KQML or FIPA ACL. Although this is a circular definition, it captures essential elements of agency: agents need not know about the internals and the implementation of other agents (they only need to know and understand the domain) and they communicate with other agents in a high-level language that allows them to make statements about their internal state and the domain they interact in. This conceptualisation of agents and ontologies, and of the relationship between them, is rooted in the work of the knowledge sharing effort (Neches *et al*., 1991), that gave birth to Ontolingua (Gruber, 1993), a language for describing ontologies and an environment for the collaborative development of ontologies, and to KQML, which introduced the concept of an Agent Communication Language (Finin *et al*., 1997) and was the conceptual basis for the FIPA ACL, the cornerstone of the FIPA[1] specifications.

1.3 Enterprise application software

Enterprise application software[2] is the computing foundation for conducting business and e-business. Collectively, such applications drive the business of every major enterprise. They are extremely costly to deploy and maintain but their high cost is offset by the tangible benefits they provide. Despite overlapping between the major categories of enterprise software, ERP, CRM and SCM systems are the major categories.

Enterprise Resource Planning (ERP) An ERP is a multi-module enterprise application that supports operations such as product planning, parts purchasing, maintaining inventories, interacting with suppliers, providing customer service and tracking orders. ERPs often include application modules for the finance and human resources aspects of an enterprise. Typically, an ERP system uses, or is integrated with, a relational database system.

Customer Relationship Management (CRM) A CRM system supports marketing, sales, and customer service. Examples of tasks supported by a CRM system include: (1) enabling marketing to identify and target their customers, manage marketing campaigns and generate leads for sales, (2) assisting sales by optimising information shared by multiple employees, (3) improving customer service and satisfaction and (4) helping employees to better understand their customers and their needs.

Supply-Chain Management (SCM) SCM is the oversight of materials, information and finances as they move in a process from supplier to manufacturer to wholesaler to retailer to consumer. Supply-chain management systems coordinate and integrate these flows both within and among companies. The oft-quoted goal of modern business (and of a successful SCM system) is the "real-time enterprise", which means reducing inventory (and the costs associated with it) without sacrificing the

[1] The Foundation for Intelligent Physical Agents (`http://www.fipa.org`) is an international standardisation organisation focusing on agent technologies.
[2] ITtoolbox (`http://www.ittoolbox.com/`) is a useful resource on the "practice" of enterprise software.

ability to respond to fluctuations in supply and demand. There are two main types of SCM software: planning and execution. Supply-chain planning applications design and implement scheduling systems for enterprise systems, e.g. the best way to fill an order. Supply chain execution applications focus on SCM logistics, such as coordinating the production, transport and storage of materials, and financial information involving all affected parties.

2 Sourcing and procurement, direct versus indirect materials

Initial efforts on deploying agents in enterprise scenarios have focused on agents that engage in automated discovery, negotiation and eventually purchasing of goods, on behalf of (enterprise) users. Aside from the question of whether enterprises would let software autonomously (or semi-autonomously) transact on critical business functions, it is crucial to understand the functions of sourcing and procurement and the distinction between direct and indirect materials.

Sourcing generally deals with the search for and identification of suppliers, materials and services. Sourcing is concerned with long-term decision-making about which materials to source, from which suppliers, under what contract terms and so on. Procurement, on the other hand, generally deals with the day-to-day activities of purchasing materials. The primary goal of procurement is to ensure the uninterrupted supply of materials by purchasing under contract from current suppliers, by identifying new suppliers and by purchasing from new and existing marketplaces.

The first wave of procurement applications focused on the procurement of indirect or Maintenance, Repair and Operating (MRO) equipment, but not on direct materials. MRO materials are not related to manufacturing; they include copier toner, light bulbs, toilet paper and so on. Direct materials, on the other hand, are directly related to manufacturing and include a wide variety of product components. One of the primary distinctions between direct and indirect materials is that indirect materials do not require customisation because they can be purchased from a catalogue. Direct materials, however, may require customisation depending on the type of product, process or system being implemented. More often than not, it is hard to produce reliable, up-to-date catalogues for direct materials, due to the lack of standardised part numbers and parts (materials) features. In addition, because direct materials are critical to the manufacturing process, a variety of considerations relating to continuous quality supply have to be made, e.g. consistent high quality of parts shipped and timeliness of deliveries are usually more important than price. Still, the above should not be interpreted as a universally applicable rule or quantifiable heuristic because the weight of each "dimension," depends on complex context-specific considerations, such as the criticality of the part in the production process, existing contracts, whether a supply disruption has occurred and so on.

3 Obstacles and opportunities

Agents have to process data from, and write data back to, enterprise systems. These systems are big and complex databases[3] typically accessed through "thin" clients that trigger updates every time an action takes place. Although the schemas and metadata of these databases are available, understanding the semantics of affected data and business processes (e.g. issuing a purchase order) requires years of familiarity and experience, because these applications are complex and often extensively customised within a particular enterprise or business unit (often a business unit of an enterprise uses its own ERP or SCM systems).

The above is the domain of Enterprise Application Integration (EAI is a new term for traditional system integration). The goal of EAI is to offer bi-directional transactional integration between the back-end systems (databases, legacy applications and enterprise applications) and the front-end (usually Web-based input, or other data-entry applications) or new applications that need to access and

[3] A tidbit of anecdotal evidence: a complete installation of Oracle ERP with all available modules and only "dummy" data for some of the included tables is in he order of 80 GB (the figure does not include the database server itself).

process enterprise data. The challenge of EAI is not the plumbing of connecting to the appropriate systems and moving data between systems in real time (not a small feat by itself) but the semantic integration between data and processes across systems, especially between systems that are owned by different business units or enterprises. This is the space of opportunity for ontologies and particularly agents that rely on layers of abstraction (content, ontologies, communication language) in order to hide the complexities and the design of the interacting systems.

Ontologies are important because they can capture domain knowledge for particular tasks and problems. The realisation has already been made, as evidenced by the various efforts to define a-priori ontologies for particular domains (e.g. RosettaNet). Such efforts will take time and they are subject to the pitfalls of standardisation efforts.

Alternatively, ontologies can be thought of as continuously evolving specifications, defined collaboratively by multiple parties that have differing levels of authority on what to define and modify. Some of the desiderata of such environments are: (1) to define objects of arbitrary complexity and with multiple attributes, (2) to allow multiple parties to collaboratively specify and modify attributes, (3) to define constraints on the values of the attributes and across multiple attributes in the same specification, and (4) to track the update process of specifications (versioning) and compare versions of it. Many business problems and processes are, or can easily be described as, collaborative processes that involve many parties (within or across enterprises) where the goal is to collaboratively evolve an (initial) multi-attribute specification. The final specification of this collaboration is the completed business process. Examples of applications that can be defined in such terms are RFQ[4] engines, collaborative sourcing and supplier certification.

Alongside work on formalisms for expressing ontologies, there is a need for robust environments for collaborative specifications of ontologies that can be used for the types of problem described above. The Ontolingua server is such an example; another one is Ontobuilder,[5] which evolved from internally used tools at VerticalNet, while building exchanges for various vertical industries. Ability to import simple ontologies, for example existing hierarchies and vocabularies expressed in XML, is an important feature for such environments. Using a relational database for the permanent storage of ontology data, and fast read and write access to that data, is another. Such considerations have to be evaluated in conjunction with the representation(s) supported by the ontology-building environment.

Agents, empowered by ontologies, can integrate resources and processes across systems owned by different business units and enterprises. A fundamental problem for researchers is the lack of access to "live systems", i.e. implementations of enterprise systems like ERPs and SCMs. Adoption of agent technologies will be difficult without assurances that agents can achieve bi-directional access to the proper data and that their designers fully understand the affected processes. Furthermore, research efforts that focus only on the agent-supported task, e.g. negotiation over goods and services or information-brokering, without considering the integration problems, are at risk of over-simplifying the semantics of data and processes and thus rendering themselves unusable. For example, the assumption of universally unique and readily available part numbers is false (especially for direct materials); correctly "cleansing" and matching part numbers is by itself a huge, if mundane, problem for e-business.

Still, positive signs exist. The J2EE[6] Connector Architecture (JCA), aimed at providing a Java-based platform and architecture for connecting to enterprise applications, is gradually being supported by more enterprise software vendors. Also, the momentum towards making available back-end

[4] Request For Quote engines are at the core of systems that allow buyers to publish complex descriptions of parts or components, along with price, quantity and contractual and other requirements, in order to invite sellers to submit quotes for procuring them.

[5] `http://www.ontobuilder.org`.

[6] Java(TM) 2 Platform, Enterprise Edition (`http://java.sun.com/j2ee/`), which is similar in scope to Microsoft's.NET, is a platform-independent, Java-centric environment from Sun for developing, building and deploying Web-based enterprise applications. One of its specifications is Enterprise JavaBeans (EJB), which is a server-side component architecture for the development and deployment of distributed object systems for the Java platform.

functionality as "services" and services support through efforts and standards such WSDL,[7] UDDI[8] and SOAP,[9] will assist researchers with getting access to real data. After all, agents are partly a new methodology towards developing software; "wrapping" functionality in a system that is called an "agent" will be of no use as long as that piece of software is not integral to the environment it interacts with and transacts upon. Although I do not advocate unconditional adoption of industry efforts, it is important to avoid duplication of labour and to attempt to enhance, build upon or take advantage of industry efforts such as UDDI and WSDL, rather than attempt to introduce a specification or a protocol that addresses the same (or a similar) problem in a more sound or extensible manner.

The brief space of this presentation lends itself to a narrative of aphorisms. In that spirit, I have a last one to offer: enterprise software is still very much a database world. Still, other technologies have found a purpose within enterprise computing: data-mining, optimisation, constraint-programming and rules engines. Agents are different because they do not necessarily offer new functionality, as is the case for example when data-mining is applied against transactional data, but a different fundamental way of managing the interaction between complex systems. Agents and ontologies are the kind of fundamental technology that can drastically change the face of enterprise software but it will take time and understanding of a very challenging "real world".

References

Finin, T, Labrou, Y and Mayfield, J, 1997, "Kami as an agent communication language" in JM Bradshaw (ed) *Software Agents* AAAI Press/The MIT Press.

Genesereth, MR and Ketchpel, SP, 1994, "Software agents" *Communications of the ACM* **37**(7) 48–53.

Gruber, TR, 1993, "A translation approach to portable ontologies" *Knowledge Acquisition* **5**(2) 199–220.

Neches, R, Fikes, R, Finin, T, Gruber, T, Patil, R, Senator, T and Swartout, W, 1991, "Enabling technology for knowledge sharing" *AI Magazine* **12**(3) 36–56.

[7] Web Services Description Language (`http://www.w3.org/TR/wsdl`) is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint.

[8] Universal Description, Discovery and Integration (`http://www.uddi.org/`) is a specification for (rather simply) describing services and discovering services. UDDI servers are used as directories of services, where providers can publish their services and "consumers" can search for them.

[9] Simple Object Access Protocol (`http://www.w3.org/TR/SOAP/`) is a lightweight, XML-based protocol for exchange of information that consists of an envelope that defines a framework for describing what is in a message and how to process it, encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses.