Extreme Competition

Peter Fingar
Executive Partner
Greystone Group

pfingar@acm.org

Latest book:
Business Innovation in the Cloud: Executing on Innovation with Cloud Computing
Meghan-Kiffer Press
www.mkpress.com/BIC

Agent-Oriented BPM (aoBPM)—and a Confession

Going one step beyond business process management (BPM) technology, intelligent agents are poised to transform the way we model the enterprise and build information systems. In the first of a two-part Column we explore the what and why of this advanced technology, its business and technical benefits and implications. In part two we will explore how to apply the paradigm to business, explore architectures and standards, and explain the relationship with BPM technology.

Part 1 :
Competing for the Future with Intelligent Agents

Creative Destruction – What Came Before What's Coming Next

The world economy is going digital and a handful of business pioneers already provide exemplars of the tactics and operation of 21st century business competitors. *Though we are no longer amazed,* the amazing story of a virtual bookstore, Amazon.com teaches many lessons for the business battlefront of the future. Out of nowhere, this digital bookstore turned an industry up-side-down, using blitzkrieg tactics while most retailers feared the security risks of this newfangled thing, the Internet. What happened here was more than creating a Web site, an intelligent and ubiquitous digital business was conceived and implemented globally. The company represented a competitive form that in 1995 had not been seen before on the business landscape and was not possible prior to the ubiquitous communication network, the Internet. Shattering traditional value chains in the book-selling industry, Amazon.com opened thousands of virtual bookstores in its first months of operation. These virtual bookstores are independently operated by people and organizations (Amazon calls them Associates) that create Web sites in their specialty fields and seamlessly interoperate with Amazon.com.

The many tasks needed to establish a virtual bookstore are handled digitally. Once the bookstore is established, "software agents" (dubbed *Eyes and Ears*) notify the virtual bookstore operator of relevant new books, and thousands of activity and sales reports are distributed to Associates each week without a drop of ink. Meanwhile, digital sales clerks gently suggest to customers browsing a particular book that others who have bought the book also bought these other titles … a little cross-selling to the self-service customer. True to the notion of the virtual corporation,

automating very different *kinds of things*: business processes and workflows which are human, knowledge-based phenomena; and opening our core business systems for direct interaction with customers, suppliers, and sometimes with competitors (i.e. the airlines industry). New forms of pervasive communications have allowed forward thinking business people to blur industry boundaries and create virtual corporations whose core competencies are information and knowledge management, taking much of the operational information once in workers' heads and codifying it into executable software.

The new realities of business have created new imperatives for business information systems. Today's business systems must provide enterprise (and inter-enterprise) reach so that islands of disparate information can be integrated into a meaningful whole. They must be able to cope with the overwhelming complexity of distributed technology and an inter-enterprise information base. They must be open to survive a network-centric ecosystem. Rapid applications development goes without saying, and applications must be designed to embrace constant change. Business systems must be knowledge-based (not just information-based) if they are to cope with the incompleteness and ambiguity of real business processes and workflows. And they must be adaptive to meet the needs of the moment and bring productivity to an increasingly overwhelmed business user and self-service to our customers. That's what is being asked of IS today, a very tall order.

Both BPM technology and agent paradigms focus on addressing *change* and *complexity*. Intelligent agent technology is the next logical step in moving the BPM technology paradigm forward and overcoming some of its shortcomings. Both technologies have been around for quite a while but only now are they being repurposed for business in response to a rapidly changing world.

While the retail industry was slow to grasp the significance of the Internet—until it got *Amazoned*, today we are witnessing a seismic shift in information technology. It's called the Cloud, and cloud computing is set to unleash a perfect storm in business, especially in today's dire economy. But unlike the retail industry that got *Amazoned*, the Cloud will affect *every industry* and go beyond enabling a virtual company and on to multi-company virtual business networks and ecosystems. Complexity will grow exponentially, and tools and methods are desperately needed to build businesses that are complex adaptive systems capable of bringing order out of the chaos (charodic systems).

Today businesses are grappling with how to make software accommodate *change* and thrive in the *complex* Cloud economy. Business leaders know all about rapid change and increasing complexity, and some are now asking "Why can't software change itself to keep up with the changes in the business?" It is this business case that is bringing intelligent agent technology out of the research lab and military establishment and repurposing it for competitive advantage.

Software Agents

Few in IT have not heard of software agents. Thanks to the dramatic growth of the Web, computing literature is replete with discussions of software agents. Like technology terms, however, the term is weary from overuse and misuse. Various names have been used - knowbots, softbots, personal assistants, software agents, intelligent agents. And various layman definitions have been offered: a software thinks, software with a head, and a smart computer program.

Backing away from technology for a moment, the everyday term, agent, provides a starting definition: "one who acts for, or in the place of, another." *A software agent is a software package that carries out tasks for others, autonomously without being controlled by its master once the tasks have been delegated. The "others" may be human users, business processes, workflows or applications.* If you use a word processor such as Microsoft Word, you are a user of autonomous

agent technology—the spelling and grammar checkers are carrying out those tasks for you, autonomously, as you type!

Is it a software agent or just another program? A *basic software agent* stands on three pillars, three essential properties: *autonomy, reactivity, and communication ability.* The notion of autonomy means that an agent exercises exclusive control over its own actions and state. Reactivity means sensing or perceiving change in their environment and responding. And, even the most basic software agents have the ability to communicate with other entities: human users, other software agents, or objects.

Add to this definition the ability to plan and set goals, to maintain belief models (their own and other agents' beliefs), to reason about the actions of itself and other agents (including humans), and the ability to improve its knowledge and performance through learning, you then have the core ingredients of an "intelligent agent." An intelligent agent represents a distinct category of software that incorporates local knowledge about its own and other agents' tasks and resources, allowing it to operate autonomously or as a part of a community of co-operative problem solvers (including human users), each agent having its own *roles* and *responsibilities*.

Figure 1.1 shows the anatomy of an intelligent agent. An agent has both "transient" (the active workspace) and "permanent" knowledge (permanent means intrinsic in this context; it stays stable from one execution cycle to the next). It has a controller (representing its head) which can be given alternative behavioral characteristics (e.g. stimulus-response- like, actor-like, or blackboard-like). An *intelligent* agent performs tasks that have both declarative and procedural components. It can use alternative reasoning strategies, including belief management. An agent's repertoire of tasks represents its capabilities. Each task can have its procedural "how to do" component represented as rules, knowledge sources (rule sets), or methods.

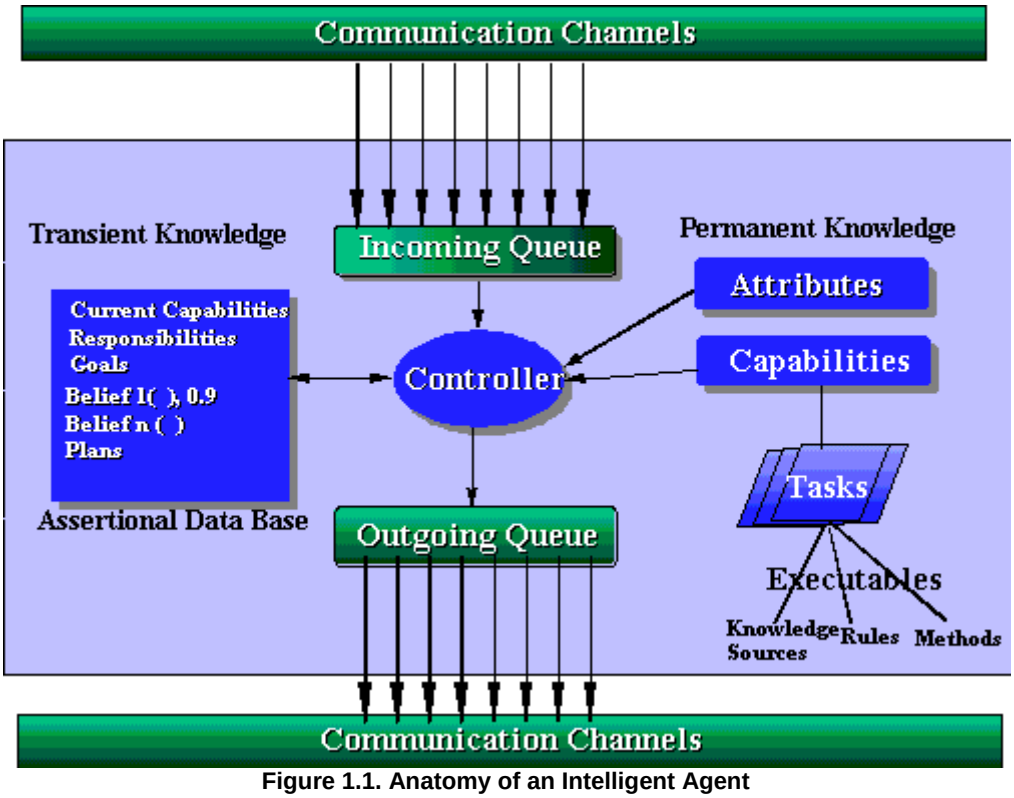


Figure 1.1. Anatomy of an Intelligent Agent

Figure 1.2 shows some of the key components of multi-agent problem-solving. To achieve common goals agents need coordination. Effective coordination requires cooperation, which in

turn can be achieved through communication and organization. So in the process of developing agent-based systems we need answers to several fundamental questions:

- What are appropriate agent architectures for different classes of problem-solving?
- How does an agent acquire its knowledge and how should it be represented?
- How does a complex task get decomposed and allocated to different agents?
- How should agents cooperate and communicate with each other?
- Can an intelligent agent be trusted?

Figure 1.2. Multi-Agent Problem-Solving

Many popular software agents bill themselves as intelligent when in fact they are basic software agents. Intelligent agent technology is a sub-field of Distributed Artificial Intelligence (DAI). DAI is concerned with issues arising out of problem-solving by a collection of smart entities/systems and concurrency of computation at different levels. Some of the basic tenets of problem-solving by intelligent agents are:

- Each agent may have different knowledge, capabilities, reliability, resources, responsibilities or authority.
- Different agents may perceive the same event or object differently.
- Agents may specialize in or focus on different problems and sub-problems.
- An important goal is convergence on solutions despite incomplete or inconsistent knowledge or data.

What is an "object?" Prior to the notion of object-oriented programming, computer programs contained procedures or code, but the data they operated on were totally separate from and outside the program (Fortran, COBOL, etc.) Objects encapsulate data and the operations on them so that only the operations are publicly accessible via an interface and internal details of the data structures are hidden. Other programs call on the object with a request and the object responds, but the requestor has no access to the internals of how the object does what it does. This request-respond approach greatly simplifies the construction of large, complex programs and is the basis of most contemporary computer programming languages (e.g., C++, Java). Think of objects as actors on a stage, calling upon one another to get things done. Object-oriented programming emerged in the 1960s (Simula and Smalltalk) and has advanced through many phases including distributed object computing; and the granularity has increased from primitive

objects to components that package business services; and the request-respond interface has moved on to Web services, and now on to Cloud services.

Intelligent agents are similar to object-oriented software (objects) in a number of ways. Agents can be organized into generalization and specialization hierarchies, to exploit *inheritance* (e.g., a risk assessment agent represents a specialization of a personal assistant agent class). An agent can advertise its services using a variety of means, and how it implements these services should be transparent (i.e., encapsulation of behavior). Different agents can respond to the same service request differently without the requester needing to know about such differences. For example, imagine that a user is supported by a set of E-mail agents (for Internet mail, LAN mail, etc). This user should not have to be concerned with how each agent handles the same request for sending a message (i.e., *polymorphism*). And, intelligent agents communicate asynchronously by message passing, using a variety of rich interaction protocols (e.g., negotiation and conflict resolution protocols).
Intelligent agents differ from software objects in a number of ways. Intelligent agents manipulate objects to perform their tasks. Behavior of intelligent agents (the tasks that they perform and how the tasks are performed) can be modified dynamically, due to learning or influence of other agents. Intelligent agents can be autonomous, can reason about themselves and can be mobile. Intelligent agents can actively and dynamically seek to cooperate to solve problems, using task and domain-level protocols (an important goal is convergence on solutions despite incomplete or inconsistent knowledge or data).

Business objects make a major contribution to modeling information in the enterprise while BPM modeling methods are great for modeling work processes in an organization. Intelligent business agents extend this capability to provide the breakthrough in modeling knowledge in the enterprise.

A major reason for the apparent failure of object-oriented (OO) and BPM approaches to deliver on reuse is insufficient attention to the issue of domain understanding and the representation of this understanding in an unambiguous and precise way so it can be validated by business people who are domain experts. Arguably, both OO and BPM methods focus too much on notation (UML and BPMN) and not enough on content. OO notions such as class, association, and message are too biased toward the implementation of software systems. Such concepts are not sufficiently expressive of business concepts: rules, constraints, goals and objectives, and roles and responsibilities. Nor are these concepts readily understandable by business people, posing a major barrier to the participation of business users in validating OO models. The lack of such validation can have serious consequences for the validity of the software solutions that are later implemented.

The agent-oriented approach provides a more *understandable* paradigm for constructing models of a business domain, as well as offering techniques for creating smart business solutions—all enabled, but not driven, by object technology. The agent-oriented approach to business objects distinguishes between active, goal-seeking objects (agents) and passive objects. Agents are the centerpiece in domain modeling, together with the tasks they perform, how they interact with each other, how their actions generate events, and how they are organized around and manage business processes. It is advocated that the same perspective and vocabulary be used during analysis, design, and construction of software systems guided by reference architectures. This method of software construction helps reduce cognitive dissonance through the use of constructs in software that can be mapped more naturally and directly to real-world entities than basic objects that underlie BPMN process models at the implementation level. In addition, agents can be integrated frameworks that contain, in one package, specific problem-solving functions, data and control. This helps to transform the creation of applications into a high-level task of assembling pre-fabricated frameworks, as opposed to the more complex task of grappling with object libraries or fine-grained components.

Going beyond today's business objects, Web services and Cloud services, *intelligent business agents* are the next higher level of abstraction in model-based solutions to business problems. By building on the distributed object foundation, intelligent agent technology can help bridge the remaining gap between flexible design and useable applications. Intelligent agents support a natural merging of object orientation and knowledge-based technologies. Intelligent agents can facilitate the incorporation of reasoning capabilities within the application logic (e.g. encapsulation of business rules within agents or modeled organizations). They permit the inclusion of learning and self-improvement capabilities at both infrastructure (adaptive routing) and application (adaptive user interfaces) levels. Intelligent user interfaces (supporting *task centered* user interfaces and intelligent assistance to end-users) can be a boon to productivity in a network-centric world.

On the technology side, intelligent agents solve run-time operations bottlenecks inherent in network-centric computing. Management of enterprise-wide objects in a run-time environment becomes more difficult using pre-programmed control objects. For example, as the size of an distributed system grows, the number of messages between objects grows non-linearly and "controller objects" themselves need to be coordinated or they can become performance bottlenecks. Unlike objects, intelligent agents can participate in high-level (task oriented) dialogues through the use of interaction protocols in conjunction with built-in organizational knowledge. In many cases, the need for communication is greatly reduced, as within these high-level dialogues, complex packets of procedural and declarative knowledge as well as state information may be exchanged in the form of mobile agents.

Figure 1.3 shows an example of an intelligent agent specializing in supporting risk management. It shows that such an agent would use its knowledge of risk assessment, its role in supporting this process and historical data to examine various data feeds in order to compose a risk exposure picture.

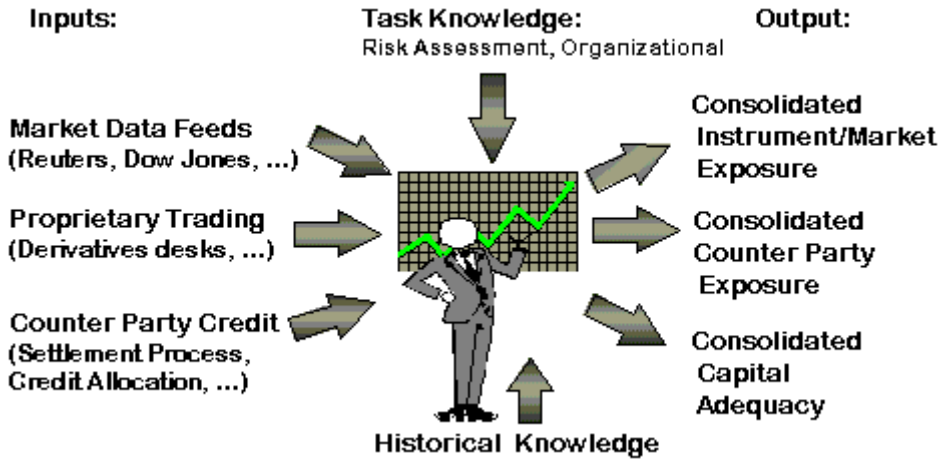


Figure 1.3. Risk Management Application of Intelligent Agents

Figure 1.4 illustrates a community of intelligent agents, including a Risk Assessment Agent, supporting a financial trader.

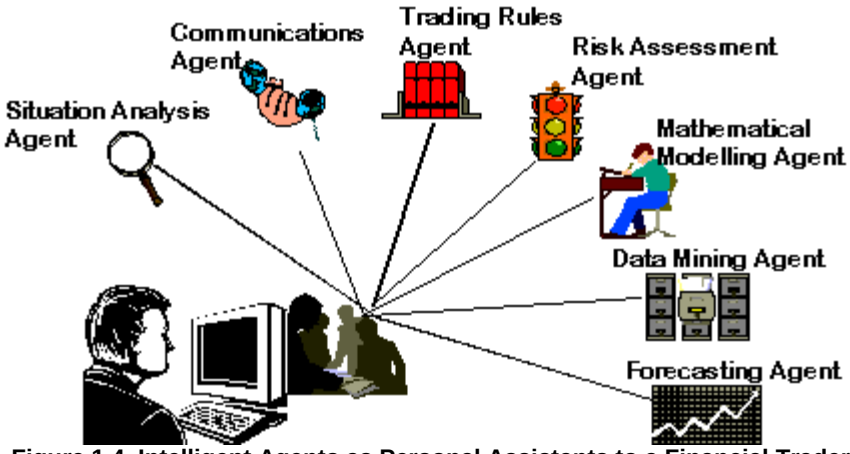


Figure 1.4. Intelligent Agents as Personal Assistants to a Financial Trader

Competing for the Future

The importance of agent technology is now widely recognized in several industries, especially defense, telecommunications, and manufacturing, as well as standards bodies such as the Object Management Group. These and other leading organizations have recognized the business and technical benefits:

1. increased productivity of business computer users and self-service customers in a network-centric business;
2. a combined information and knowledge model of the business and information systems that makes rapid development in a complex environment a reality; and
3. bandwidth efficiency so necessary in a network-centric computing environment.

Not only do they recognize the benefits, some companies are already deploying intelligent agent technology as a competitive weapon--today! They turn Web hits into business relationships. They use smart technology to plug suppliers and customers into their core business systems. They profile their constituents and search for new patterns in their data warehouses to better meet ever changing needs. Is your company ready to complete with them?

With careful planning and armed with a solid understanding of the current limitations of intelligent agent technology, powerful business information systems can be developed that exploit the Internet and distributed objects to gain strategic business advantage. Although the tasks of designing and building them are not trivial, information systems based on intelligent agent technology are inevitable. Learning to harness this emerging technology should be a top priority of today's business and technology leaders. Companies can go it alone but those that face immediate mission critical requirements will be well advised to seek outside assistance from those who have gone before.

# # #

Part 2
Developing Enterprise Systems with Intelligent Agent Technology

Going one step beyond distributed object technology, intelligent agents are poised to transform the way we model the enterprise, build information systems and manage business processes. In the first of this two-part article we explored the what and why of this advanced technology, and its business and technical benefits and implications. In part two we explore how to apply the paradigm to business, explore architectures and standards, and explain the relationship with object and BPM technology. An in-depth exploration of this topic could fill a few books. This article aims to describe the terrain and to provide a road map of the territory.

We concluded in the first part of this article that information systems based on intelligent agent technology are inevitable, even though the tasks of conceiving, designing and building them are not trivial. As we turn our attention to these tasks, we will benefit from keeping the fundamental question in mind, "How can we conceive, design, and build adaptive, multi-enterprise-scale information systems and business processes?" Intelligent agent technology is an enabler of such systems, but is not an end unto itself. When combined with the maturing disciplines of object technology and BPM methods, their synergy, embodied in *agent-oriented BPM* (aoBPM), can take us to a new level in the search for the prize of managing complex business ecosystems and multi-company business networks.

Our approach to building agent-based systems can be characterized in terms of three distinct, but related, topics:

1. *A full-lifecycle solution development process* that explicitly addresses domain modeling—the effort spent in domain and problem understanding can be leveraged to guide and accelerate information systems development.
2. *Ontology-based domain models*—fully understanding the domain is essential to understanding the problems in the domain. The richness of domain models determines reuse and flexibility.
3. *Architecture*—agents must follow a consistent architectural approach so they can interoperate with each other.

After a brief introduction to the agent-oriented lifecycle, we focus our attention on the heart of the matter, domain modeling and architecture.

A full-lifecycle solution development process

The agent-oriented lifecycle model should be iterative and incremental. Over a decade of experience taught us that agent orientation will enrich contemporary OO and BPM approaches by providing an *active* modeling paradigm and better analysis models that enable reuse. As shown in Figure 2.1, our agent-oriented domain modeling and architectural frameworks can be integrated with best practice software engineering lifecycles, leveraging the contributions of artificial intelligence (AI) and object-oriented software engineering. We thus preserve considerable intellectual capital while taking the next step in development methods for truly intelligent and adaptive information systems. Radical new methods will not be accepted as corporations are not in this to advance computer science, they are in business.

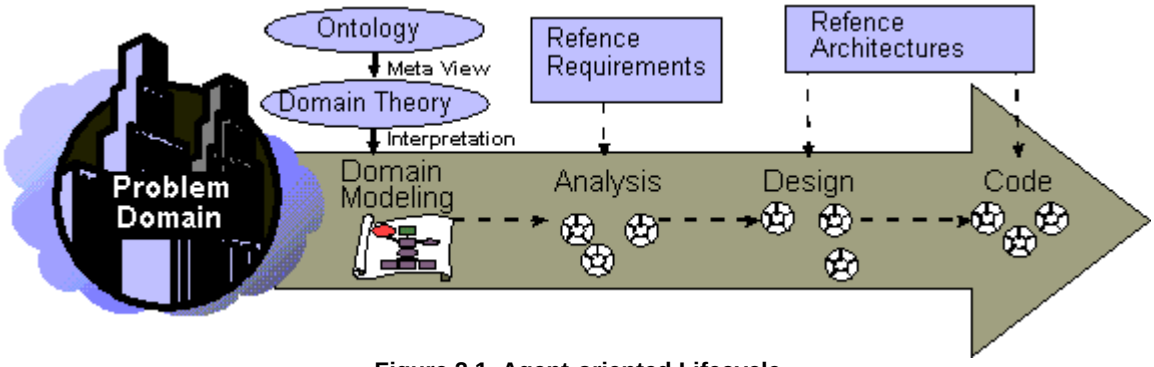


Figure 2.1. Agent-oriented Lifecycle

Ontology-based domain models

Our approach to building agent-based solutions emphasizes the importance of domain modeling as a critical initial step in producing business technology solutions. There are at least three distinct approaches to domain modeling: business process management (BPM), object-oriented (OO), and artificial intelligence (AI) which the underpinning of intelligent agent technology. All three approaches are model-based and offer techniques for describing problem domains. As shown in Figure 2.2, the three approaches offer differing perspectives of the problem domain. The convergence of OO, BPM, and AI methods results in a significant breakthrough in building models of the enterprise that are capable of end-to-end integration of business analysis and software systems. Although the fusion of these disciplines is not yet complete to the point of standardization, pioneering companies have recognized the competitive advantage and launched major agent-oriented development initiatives.

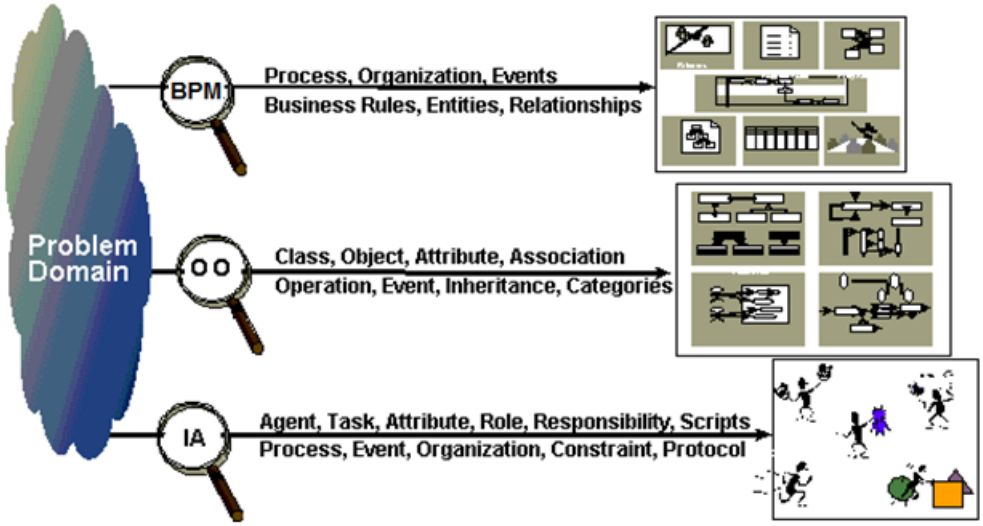


Figure 2.2. BPM, OO and IA Perspectives on Business Domain Modeling

Traditional BPM methods are expressive and easily (often, naively) understood for describing business process and organization. But traditional BPM produces separate models of data, processes and organization that soon become unmanageable. The models are difficult to integrate, validate or leverage for the development of software systems. Information is captured using informal techniques even though more formal notations such as BPMN are deployed, and incomplete lifecycle models lack a clear implementation path.

Underlying high-level process models, object-oriented methods are well suited to software engineering and have potential for reuse. They are, however, not inherently business oriented,

not sufficiently expressive of most problem domains, and often foster premature commitments to design and implementation strategies. Consequently, they do not capture a sufficiently rich domain model for significant reuse. In practice, most OO methods are informal and not repeatable. Traceability is questionable and robust design heuristics are hard to find.
Business models must make a distinction between passive and *active, goal-seeking* objects (intelligent agents) if we are to close the gap between model and reality. Business object and process models need to contain behavioral declarations, role definitions, ontological awareness and constraint axioms.
Intelligent agent technology can be leveraged to enhance enterprise modeling as well as offering new techniques for developing intelligent applications and smart technical infrastructure services. An agent-oriented perspective allows us to develop rich and expressive models of the enterprise as the foundation for adaptive, reusable business software.

An agent-oriented approach such as the Knowledge Analysis and Documentation System (KADS) supports richer representation of the problem domain and tend to be more rigorous and mature compared to BPM and OO. A method like KADS entails little commitment to design in early modeling phases and allows alternative implementation paths. On the down side, methods like KADS are not widely known outside the artificial intelligence community. They tend to be weaker on infrastructure issues and lack the diversity of commercial tools found in the mainstream technology world. Traditional artificial intelligence is knowledge-base centric and can result in monolithic design.

What's new under the sun? Object-orientation and artificial intelligence have been around for decades. And, many informally describe "BPM" as new clothes on industrial systems engineering. It is the melding of these disciplines into a unified approach to enterprise modeling that represents the breakthrough. Blending the strengths and sifting out the weakness of each discipline, agent-oriented BPM can take us one great step forward. aoBPM is a full life-cycle proposition, not just a programming or implementation issue. The secret to aoBPM's success is to apply agent technology up front as a *domain modeling metaphor* while using the maturity of object-oriented methods to provide the infrastructure lacking in the artificial intelligence world. Figure 2.3 shows that agent-based domain modeling produces the business object model in terms of ontologies, and uses ideas and techniques from traditional business analysis (Process Modeling/BPMN), OO (UML), and AI (Agents and Rules). While devoid of implementation details, the business object model is agent-based and is used to support user task analysis, requirements modeling, and the specification and design of the software object model. The flow of activities in the figure would be somewhat different if we were working in a green field. In the real world of business, green fields are rare, and the figure reflects the use of preexisting BPM and OO analysis models.

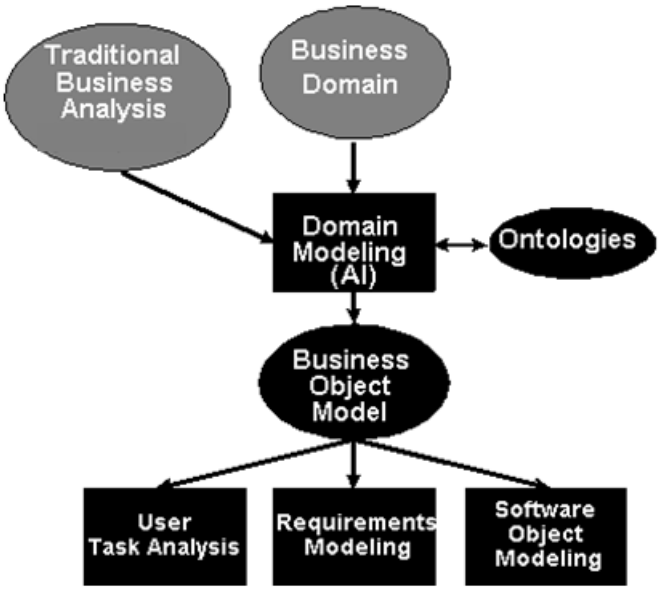


Figure 2.3. Combined BPM, OO, IA Domain Modeling Process

A domain model is commonly defined as a representation of entities, their attributes, their behaviors, and the processes that bind them together within a domain—nothing new to the experienced business analyst. What is new and highly beneficial is to define the information in a domain model in terms of ontologies and then to use these to construct business object models. What is an *ontology?* Tom Gruber of Stanford University provides the short answer, "An ontology is an explicit specification of a conceptualization." He explains, "A body of formally represented knowledge is based on *conceptualization:* the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them." An ontology defines the basic terms and relations comprising the vocabulary of a topic area, as well as the rules for combining terms and relations to define extensions to the vocabulary. All domain models embody an ontology—albeit mostly implicitly. Our agent-oriented approach to defining business objects emphasizes the use of explicit ontologies as implementation-neutral representations of knowledge that can then be mechanically translated into different target modeling tools.

An ontology can be expressed in a variety of ways, for example: informally (Natural Language, Graphical Notations), semi-formally (for example, Ontolingua), formally (first order logic languages such as Knowledge Interchange Format [KIF]). The *de facto* language for encoding ontologies is Stanford University's *Ontolingua,* a portable language for writing ontologies. Ontolingua is a LISP-like language that is based on KIF. Because KIF provides for the representation of knowledge about the representation of knowledge, storing business knowledge in KIF opens up many opportunities for delivering reuse and validation at the knowledge level.

Architecture is the Key to Agent-Oriented Development

Agent technology does not provide immunity to bad software engineering. Building enterprise-scale agent-oriented systems requires an architectural approach with a strong emphasis on software engineering processes and methods. Our approach to architecture for multi-agent systems is based on a consideration of fundamental design variables that can be instantiated in different ways to accommodate the creation of agents with reactive, deliberative or hybrid behaviors. Our approach to architecting distributed, multi-agent systems can be summarized as follows:

Given: A set of goals, objectives and tasks to be achieved or performed by a community of agents and a set of design variables,
Find: Appropriate instantiations for each design variable, based on available options, subject to applicable hard and soft constraints.

High-level design variables include:

- *Internal architecture of agents*—how each agent is constructed internally
- *Control and Coordination approach*—how intra and inter agent activities and processes are managed
- *Cooperation style*—whether and how agents work together to succeed in common goals
- *Communication, interaction and knowledge sharing approach*—how agents exchange and share information
- *Granularity of agents*—covering both process granularity and task granularity
- *Depth of adoption of agency*—starting from the user-interface, through application services and infrastructure services, to what extent should agents be used?
- *Distribution model*—the method of distributing application functionality among agents which run across multiple processors.

The following constraints are typical in enterprise systems: information bandwidths, limits of agent processing, resource availability, need for varied but guaranteed response times, reliability, security, distributability, scalability, portability.

By examining the characteristics and capabilities of intelligent agents, we can identify many of the architectural components that are needed to support an agent-oriented environment. For handy reference, Figure 2.4 repeats the illustration of the anatomy of an intelligent agent to center our discussion of architectural requirements.

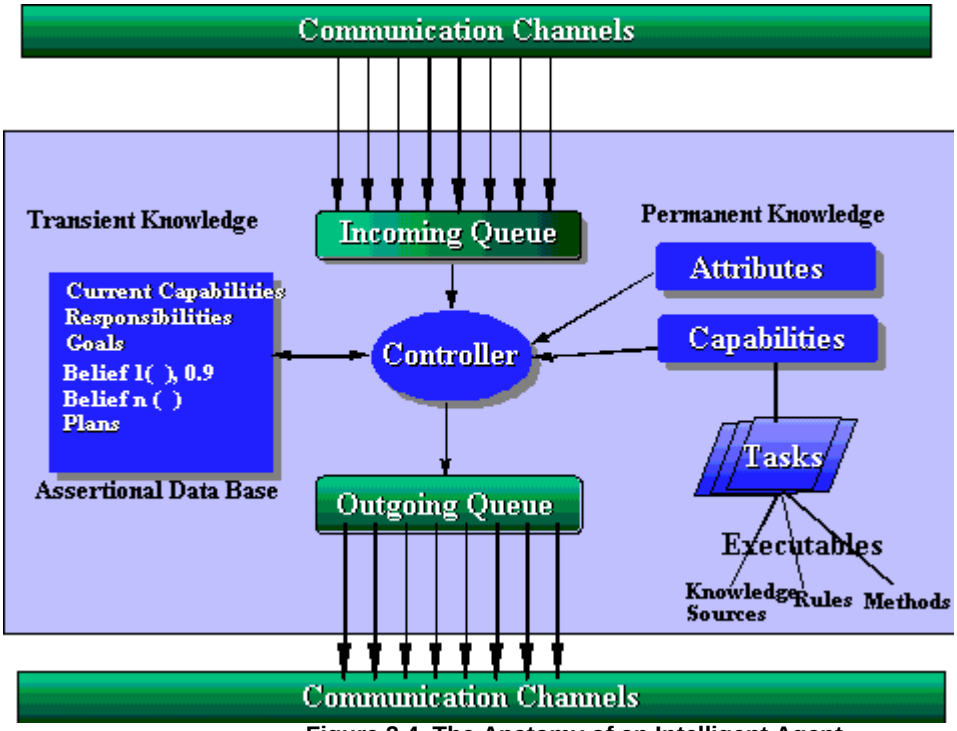


Figure 2.4. The Anatomy of an Intelligent Agent.

An intelligent agent can:

- send and receive information to and from other agents using appropriate protocols (sensing and communication) —this requires communication services that support message queue management, asynchronous messaging and content-based routing of messages.
- generate multiple objectives, goals and plans for itself and other agents, process information received and perform reasoning (e.g. inferencing, synthesis and analysis) — these require services that support rich knowledge representation schemes (e.g. for Rules and Objects), alternative logics (e.g. fuzzy logic, temporal logic, bayesian logic), alternative inferencing strategies (forward, backward, opportunistic), planning (plan generation, plan execution, and plan repair), and concurrent operations.
- maintain explicit belief models of itself and other agents and can reason with incomplete, inconsistent and uncertain information. This requires support for belief revision (e.g., Truth Maintenance).
- have a set of capabilities (which can change dynamically) —i.e. the tasks it can perform—and can reason about its own and other agents capabilities and skills (e.g., planning, inferencing, communication and negotiation). This requires a task specification language, and mechanisms that support 'learning' (e.g., Explanation-based learning, Induction)
- assume roles and perform tasks and execute physical and non-physical actions - actions can result in events which in turn may trigger other actions and processes (in this context a process is analogous to a plan and consists of a pre-defined, temporally ordered, set of activities which are designed to achieve a given goal). This requires task selection and task execution mechanisms.
- engage in complex interactions with each other, such as negotiation and task delegation. This requires support for interaction protocols (e.g. Contracting, Conflict Resolution, Resource Allocation, Basic Communication). Knowledge Query Manipulation Language (KQML) is the de facto standard for designing inter-agent interaction protocols.
- dynamically join or leave groupings or organizations - organization may be functional or social. This requires support for specification of organizational structure, organizational procedures/rules, and roles/responsibilities.
- have its own internal architecture (e.g. Blackboard, Actor or Stimulus/Response). This requires different architectural specifications and alternative control algorithms.
- explain its reasoning in terms of 'how' and 'why'. This requires explanation mechanisms.

The kinds of agents in multi-agent systems strongly influence architecture. Four major types of agents are central to large-scale enterprise systems:

- System agents—these agents are responsible for providing infrastructure-level services to problem-solving and application agents (e.g., access to operating system services, access to data repositories, communications management and security monitoring).
- Functional Units (FUNs) —FUNs (Farhoodi 1991, 1993, 1994) are conceptual agents that represent an organization unit. They define a number of global characteristics for a grouping of cooperating agents, (e.g. common goals and objectives; organizational procedures and roles and responsibilities). Important benefits of organization include structuring communication and reducing uncertainty, increasing system coherence. Figure 2.5 illustrates the relationships between FUNs, agents, roles and tasks.

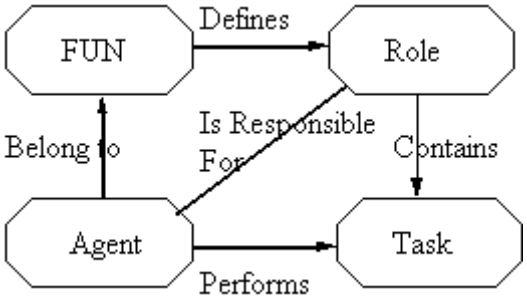


Figure 2.5. Relationships between a FUN, agents, roles and tasks.

- Application agents—these agents embody functionality or application logic within a domain (e.g., risk management). An application can be viewed as a FUN consisting of a cohesive set of agents which share information and cooperate to provide services to their clients.
- Interface agents—these mediate between application agents and external users of their services. Two sub-types of Interface Agents are: User Interface Agents and Wrapper Agents. Human users interact with the system via User Interface Agents which orchestrate other agents and FUNs to deliver user-level services. External computer systems are linked-in via wrapper agents which encapsulate their APIs behind standard protocols.

Agents are constructed from classes within three distinct hierarchies: *Controller Class Hierarchy*–contains prototype architectural behaviors (for example, Blackboard, Actor, Stimulus/Response); *Agent Class Hierarchy*–contains prototype agent behaviors, multiple inheritance being supported (for example, epistemic agent, mobile agent, social agent); and *Task Class Hierarchy*–contains default task definitions (for example, interaction management, problem-solving tasks). An agent represents a logical entity that physically comprises a cluster of components (each component type can be implemented as a separate Class of object). Such agents can be implemented in a variety of ways. For example an agent may be implemented using two frames (an Object-based AI data structure), a base frame and a Meta frame. The controller can be placed in the Meta-frame and the attributes and the assertional database can be placed or be referenced in the base-frame. Alternatively, an object-oriented approach may be used where each component is defined in its own class. The process that loads agents assembles their components and creates instances at run-time.

Agents communicate with other agents through one or more communications channels which are linked to mailboxes or message queues. Figure 2.6 shows a logical view of a system architecture that allows agents to interoperate. The post offices in the figure provide delivery services for the mailboxes. This abstraction of communication services allows alternative standards and delivery technologies to be deployed. Each agent has a knowledge-base consisting of non-executable (attributes and own capabilities) and executable (how to perform tasks) knowledge. In addition, each agent has an internal assertional database for storing dynamic information, such as its beliefs about other agents. Local services support internal operations of agents (e.g., Message-Queue Management, Query Management and Belief Management). The architecture can be physically distributed across many threads, processes or processors, depending on requirements.

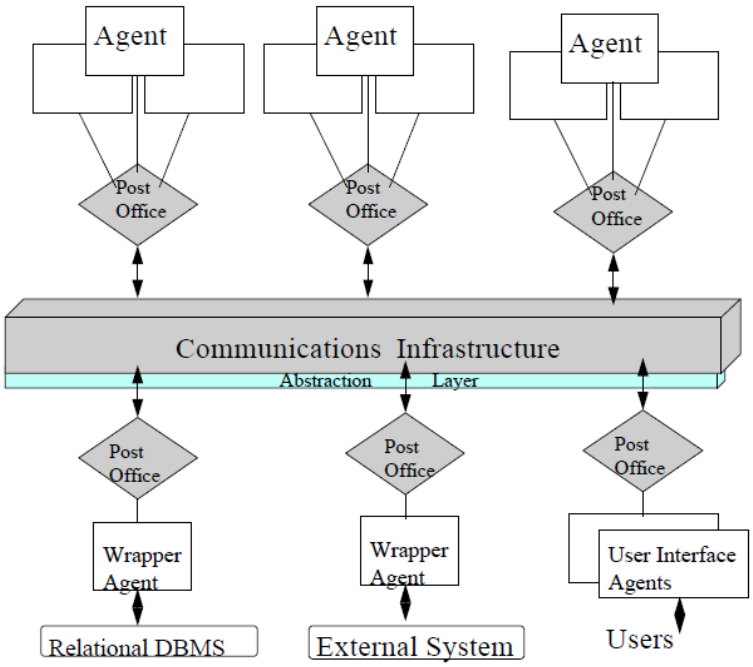


Figure 2.6. Logical View of the System-Wide Architecture

The figure highlights the way individual agents connect to communication services and the underlying technical infrastructure. The figure also shows how 'external' applications or legacy systems such as data repositories can be wrapped to appear as agents—agent wrappers are more sophisticated than object wrappers. They maintain rich semantic models of the application, database, or systems they represents. Abstraction layers insulate the agents from needing to know specific operating system or middleware APIs. The infrastructure should provide a common set of interfaces through which agents may plug into operating system services.

Although building agent-based systems will be greatly enhanced by standards-based services, these systems can be built today if a solid systems architecture is defined. Components of the architecture can be superseded with standards-compliant components as they become available. Corporations in highly competitive industries cannot wait for full standardization as their competitors are not waiting.

Conclusion for Part 2

Based on 12 years' experience gained from large-scale agent-based projects, this article described an approach to developing complex agent-based software systems. A holistic, *agent-oriented* view of design was advocated, and an appropriate lifecycle model was introduced. The discussion of implementation strategies identified a number of technical issues. Currently, a mix and match software strategy has to be adopted, resulting in the need for well defined systems architecture to maintain coherence.

Corporations will not take the step to agent-oriented BPM because they can, they will do so because they must. Now is the time to move from the current OO methods that deliver excellent results for expert programmers creating distributed systems that underlie high-level BPMN models, to the ubiquity of business process management where processes are developed by unsophisticated programmers and business people using visual development tools. While this vision of simplicity is appealing, the underlying enterprise framework is extremely complex and simply needs intelligent agent technology to manage and hide the complexity. With careful planning, sound software engineering and solid systems architecture, an intelligent greenhouse can be architected to grow the software of the future—today.

Confession and Final Conclusions

First the confession.
This Column was written 15 years ago by me and Faramarz Farhoodi (formerly with Logica, UK, where he had extensive experience with the distributed intelligent agent system, CADDIE) as a two part article for *Distributed Object Computing* Magazine (I did change some words such as BPR to BPM). What gall!

Well, thanks to Larry Roberts and his work with GTE Data Services, I had the opportunity to work with TCP/IP (the Internet) in 1969! Our whole team was full of excitement and wonder, as described in this *must-watch* one-minute clip by futurist Arthur C. Clark in 1974, long before the Internet revolution.

http://www.youtube.com/watch?v=OIRZebE8O84

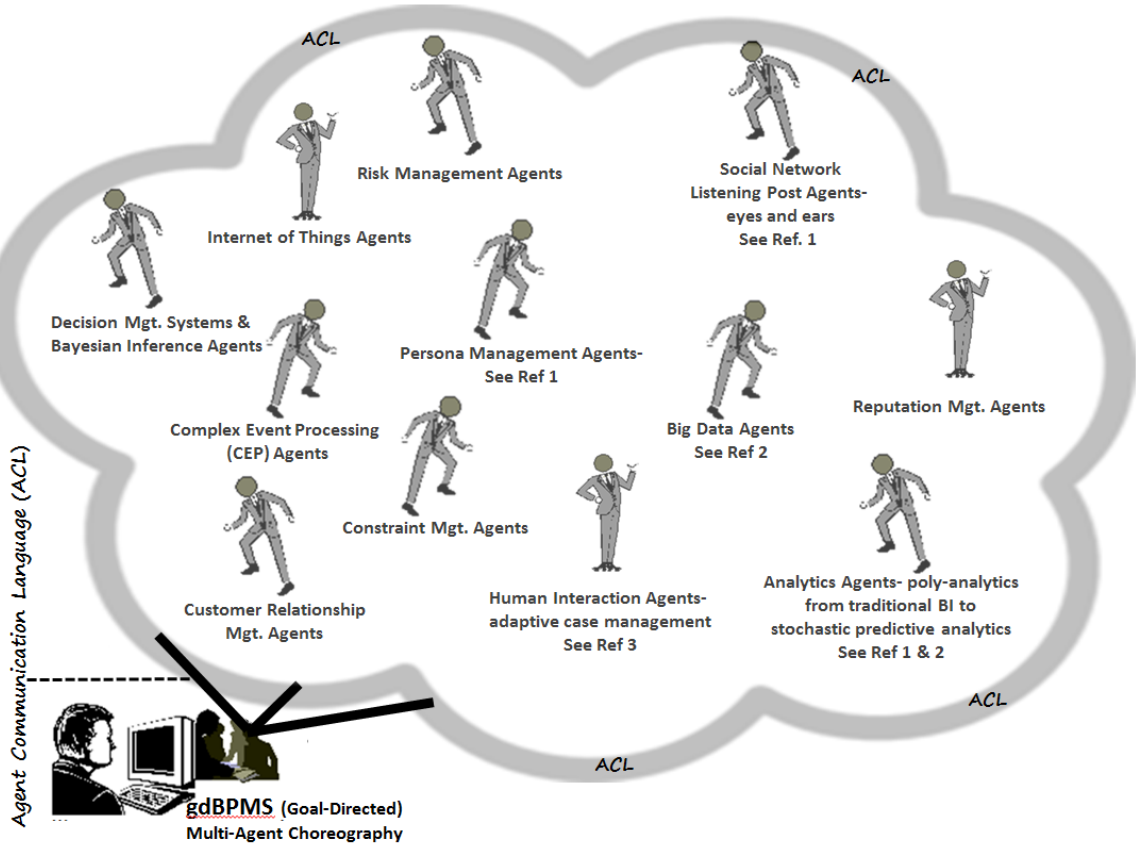
Opps. Nothing seems to happen as soon as it's "invented," so we had to wait until Jeff Bezos "Amazoned" the retail industry before the Internet became the backbone of eCommerce. Amazon is a marvel of a "virtual company," one that taps intelligent agent technology including *Eyes and Ears*, to deliver services that once before only humans could deliver.

As mentioned earlier, today we are witnessing a seismic shift in information technology. It's called the Cloud, and cloud computing is set to unleash a perfect storm in business, especially in today's dire economy. But unlike the retail industry that got Amazoned, the Cloud will affect every industry and go beyond enabling a virtual company and on to multi-company virtual business networks and ecosystems. And it won't involve just a single Cloud. Just as the Internet is a network of interoperating networks of computers, the "Intercloud" will be an interoperating network of Clouds: private, public and hybrid. Complexity will grow exponentially, and tools and methods are desperately needed to build businesses that are complex adaptive systems capable of bringing order our of the chaos (charodic systems).

That's why it is now time for Intelligent Agent technology, a technology that's been around for years, to take center stage in the world of business in general, and specifically in the world of BPM, for business processes are how work got done in the horse and buggy age *and* how work gets done in the emerging Cloud economy.

With the current reality of total global competition and world-wide operation of companies, this shift will require polymorphism, not just at the software object level, but also in our enterprise architecture: "Fractal Enterprise Architecture," (See BPrends Column at http://tinyurl.com/bt3hyez).

The *goal-driven* BPM system (gdBPMS) with *choreography* (not old-fashioned process orchestration) will be at the heart of multi-agent systems that seek the common goals set by the gdBPM. An informative goal-oriented method is described in the BPTrends Column, Goal-Oriented Organization Design (GOOD). http://tinyurl.com/cehhc5g



BPTrends references for above figure:
1. Listening Posts and Persona Management Systems
http://tinyurl.com/dx8pdfj
2. Big Data
http://tinyurl.com/cjayovs
3. Human Interaction Management
http://tinyurl.com/bpe4pdy and
http://tinyurl.com/blujzyp

Think this is a bunch of hype? Think it will be years and years before such systems affect the marketplace? Think it will take quantum computing to implement? Think again, for in many ways the future is already here; it's just not evenly distributed. In 2012, the Association of Computing Machainery (ACM) conducted a conference featuring industry luminaries including Gordon Bell and Vint Cerf addressing "The Next Fifty Years," and even the summary of the luminaries is insightful: http://www.cbronline.com/news/predicting_the_next_50_years_of_computing_1 http://www.cbronline.com/news/predicting_the_next_50_years_of_computing_2

In 2009, I was invited to speak at GE Global Research's Whitney Symposium in Niskayuna, New York, a 600 acre research campus on the Mohawk river... the legacy of one Thomas Edison (You can view the PDF of the presentation, *Business Process Management and Systems Thinking,* here: http://www.peterfingar.com/GE-Symposium-Jun09.pdf and related links here: http://peterfingar.com/GE09.html).

It was a huge honor and an eye-opening experience (1,000 PhDs representing 22 disciplines in one building!). The Theme for the Symposium was The Engineering of Customer Services. Sound boring? Not so, when you consider that 80% of the U.S. GDP is not about Agriculture, not about Manufacturing, but about Services! Like IBM before it, GE is pushing hard to become a services-driven company versus just a product company, and the Symposium brought together thought leaders on services sciences, engineering and management, including speakers form IBM's legendary Thomas Watson's Research Lab, MIT, Columbia University, and think tanks from Europe.

That was 2009.

Here's a 2012 update on G.E. from the *New York Times*, "G.E. is America's largest industrial company, a producer of aircraft engines, power plant turbines, rail locomotives and medical imaging equipment. It makes the heavy-duty machinery that transports people, heats homes and powers factories, and lets doctors diagnose life-threatening diseases. G.E. resides in a different world from the consumer Internet. But the major technologies that animate Google and Facebook are also vital ingredients in the industrial Internet — tools from artificial intelligence, like machine-learning software, and vast streams of new data. In industry, the data flood comes mainly from smaller, more powerful and cheaper sensors on the equipment.

"Smarter machines, for example, can alert their human handlers when they will need maintenance, before a breakdown. It is the equivalent of preventive and personalized care for equipment, with less downtime and more output. 'These technologies are really there now, in a way that is practical and economic,' said Mark M. Little, G.E.'s senior vice president for global research. 'G.E.'s embrace of the ***industrial Internet*** is a long-term strategy. But if its optimism proves justified, the impact could be felt across the economy.'" On a related note, a hot 2012 startup, SmartThings is bringing the capabilities of the industrial Internet of Things to the consumer market (http://smartthings.com)/

Now, GE has opened a whole new research center in "Silicone Valley. More from the *New York Times (*November 23, 2012): "When Sharoda Paul finished a postdoctoral fellowship last year at the Palo Alto Research Center, she did what most of her peers do—considered a job at a big Silicon Valley company, in her case, Google. But instead, Ms. Paul, a 31-year-old expert in social computing, went to work for General Electric. Ms. Paul is one of more than 250 engineers recruited in the last year and a half to G.E.'s new software center here, in the East Bay of San Francisco. The company plans to increase that work force of computer scientists and software developers to 400, and to invest $1 billion in the center by 2015. The buildup is part of G.E's big bet on what it calls the 'industrial Internet,' bringing digital intelligence to the physical world of industry as never before. The concept of Internet-connected machines that collect data and communicate, often called the "Internet of Things," has been around for years. Information technology companies, too, are pursuing this emerging field. I.B.M. has its 'Smarter Planet' projects, while Cisco champions the 'Internet of Everything.' But G.E.'s effort, analysts say, shows that Internet-era technology is ready to sweep through the industrial economy much as the consumer Internet has transformed media, communications and advertising over the last decade."

"At Mount Sinai, G.E. has worked on optimization and modeling software that enables admitting officers to see beds and patient movements throughout the hospital, to help them more efficiently match patients and beds. Beyond that, modeling software is beginning to make predictions about likely patient admission and discharge numbers over the next several hours, based on historical

patterns at the hospital and other circumstances —say, in flu season. The software, which Mount Sinai has been trying out in recent months, acts as an intelligent assistant to admitting officers. 'It essentially says, 'Hold off, your instinct is to give this bed to that guy, but there might be a better choice,' Mr. Keathley explained. At a hospital like Mount Sinai, G.E. estimates that the optimization and modeling technologies can translate into roughly 10,000 more patients treated a year, and $120 million in savings and additional revenue over several years."[1]

I chose this kind of simple case study from G.E./ Mount Sinai to illustrate that the move to agent-oriented BPM isn't a future big bang, ERP conversion or Orwellian event. This isn't Star Wars stuff and in no way implies complicated systems or some esoteric stuff from some research lab. We are talking about practical technology that has been working for many years. It works.
Don't let the bad-news stories of seemingly exotic agent technology providers, past and present, create fear. Perhaps you've read about H.P. going after its recently acquired Autonomy Corp. for financial shenanigans, a UK firm that grew out of Cambridge university in 1995. Despite all the rancor, Autonomy pioneered the use of Bayesian inference in the intelligent agent world. It works. Autonomy counts the U.S. Department of Homeland Security, the U.S. Department of Defense, the U.S. Securities and Exchange Commission, the Olympic Games Security Committee, the UK Houses of Parliament, General Motors, Ford, GlaxoSmithKline, Philips, BMW, Nestle and Ericsson.among its forward thinking 65,000 customers - (http://www.autonomy.com/content/Autonomy/introduction)
.
So, don't panic when you hear the term aoBPM. Don't take on all the rules nor all the possible types of analytics or try to build an all-encompassing agent infrastructure. Focus on a given process where multi-agent systems agents can make a difference, as did Mount Sinai when choosing to take on just its admission process. Such beginnings can grow into a bright future in today's world of unexpected change and total global competition.

Carpe Diem.

Author

Peter Fingar is regarded as one of the original promulgators of business process management since the publication of his book coauthored with Howard Smith, Business Process Management: The Third Wave (Meghan-Kiffer Press). As a former CIO and college professor, Peter has been working at the intersection of business and technology for almost 40 years. His recent book, *Business Innovation in the Cloud* is coauthored with the Chief Innovation Officer of Dell Services. *Dot.Cloud: The 21st Century Business Platform Built on Cloud Computing* has been published in Chinese (Beijing) and Russian (Moscow) editions. He has joined forces with Jon Pyke, founder of the Workflow Management Coalition (WfMC), and Andy Mulholland, Global CTO of Capgemini, to pen the influential book, *Enterprise Cloud Computing: A Strategy Guide for Business and Technology Leaders.* Peter delivers keynote talks across the globe and is speaking this year in Asia, Europe, and the Americas (www.peterfingar.com).

BPTrends Linkedin Discussion Group

We created a BPTrends Discussion Group on Linkedin to allow our members, readers and friends to freely exchange ideas on a wide variety of BPM related topics. We encourage you to initiate a new discussion on this publication, or on other BPM related topics of interest to you, or to contribute to existing discussions. Go to Linkedin and join the **BPTrends Discussion Group.**

[1] http://www.nytimes.com/2012/11/24/technology/internet/ge-looks-to-industry-for-the-next-digital-disruption.html?hpw&_r=0