21st International Conference on:
Systems Research, Infomatics and Cybernetics

Preconference Proceedings

Knowledge Management Systems

Focus Symposium: Baden-Baden, Germany; Tuesday, August 4, 2009

Focus Symposium Chair

Jens Pohl

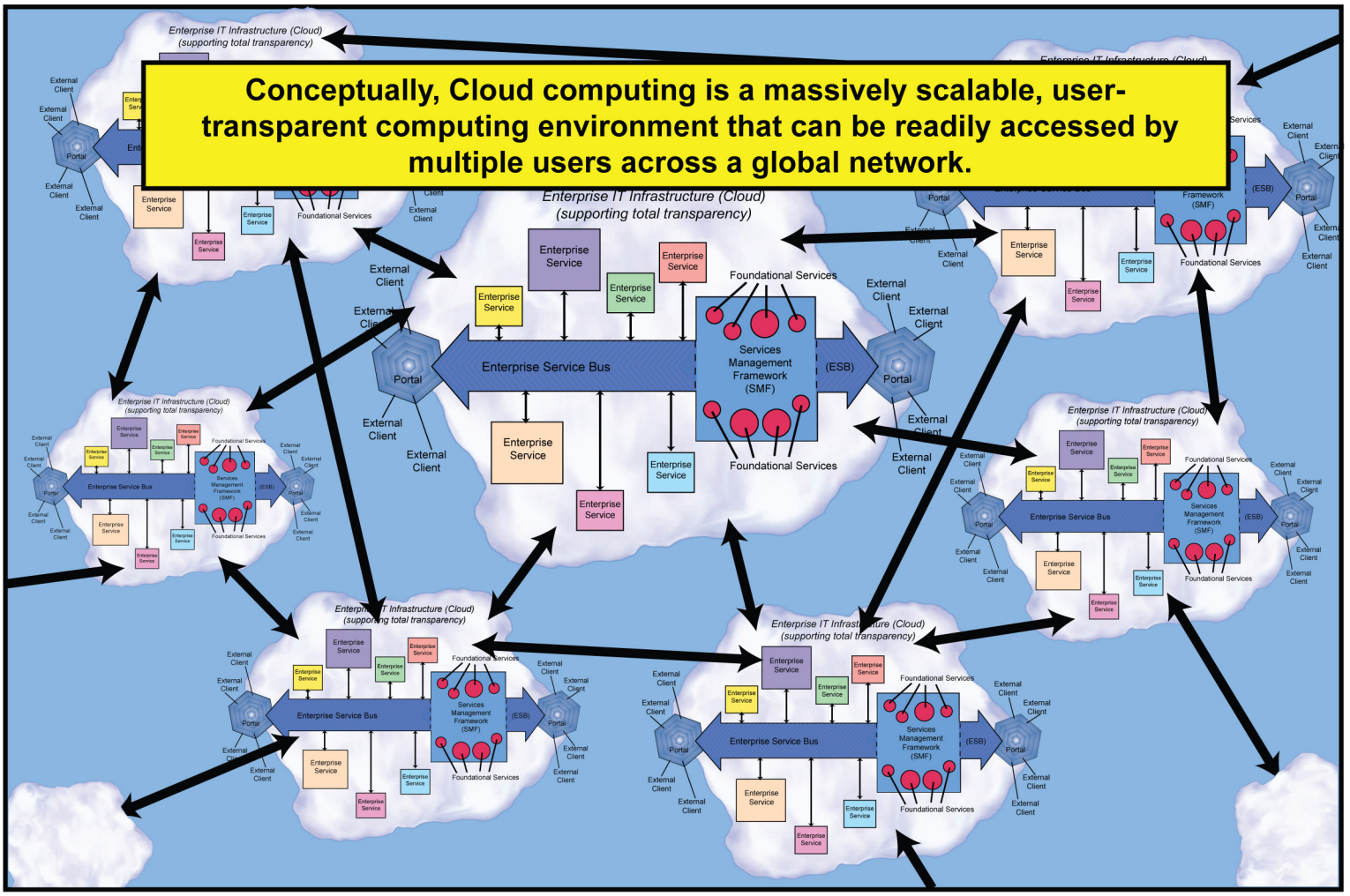
Executive Director, Collaborative Agent Design Research Center
and Professor of Architecture
California Polytechnic State University
San Luis Obispo, California, USA

Sponsored by:

The International Institute for Advanced Studies
in Systems Research and Cybernetics
and
Society for Applied Systems Research

Professor George E. Lasker
Chairman

ISBN 978-1-897546-00-0



www.cadrc.calpoly.edu

ISBN 978-1-897546-00-0

InterSymp – 2009

21st International Conference on Systems Research, Informatics and Cybernetics

(August 3-7, 2009, Baden-Baden, Germany)

Pre-Conference Proceedings of the Focus Symposium on

Knowledge Management Systems

Tuesday, August 4, 2009

Focus Symposium Chair:

Jens Pohl
Executive Director, Collaborative Agent Design Research Center
and Professor of Architecture
College of Architecture and Environmental Design
California Polytechnic State University
San Luis Obispo, California, USA

Sponsored by:

The International Institute for Advanced Studies
in Systems Research and Cybernetics
and
Society for Applied Systems Research

Professor George E. Lasker
Chairman

ISBN: 978-1-897546-00-0

Preface

One of the more subtle consequences of the rapid advances in information technology over the past several decades has been the increasing focus on the individual. Enabled by powerful communication facilities and computer-based automation tools that vastly increase the capabilities of the user, an individual person can orchestrate and achieve more today than an entire organization was able to successfully undertake a mere decade or two ago. Recognition of the value of the individual is exemplified in multiple ways, ranging from the changing structure of business corporations, the rise of entrepreneurship and self-employment, to apparently exorbitant judicial compensation awards, and the increasing value placed on human life.

Emerging out of this technology driven environment are a new set of personal values and expectations that differ significantly from past social conventions. The enabling nature of this environment, in itself, presents a challenge through the increased opportunities that it offers to the individual. To take advantage of these opportunities, the individual who is proactive and willing to take calculated risks is likely to be more successful than the individual who is reserved and conservative. Similarly, the person who is self-reliant and willing to exercise leadership to reach objectives that are based on future trends, is likely to outperform the person who is subservient and intent on duplicating past successes.

As ideas, initiative and persistent motivation become more useful human qualities, risk taking and entrepreneurship will become recognized as being increasingly rewarded and conservatism as being increasingly penalized. Under these conditions traditional values such as prudent compliance, measured reactiveness, and acceptance of the status quo will gradually fall out of favor. Instead, the more successful individual will have recognized the value of continuously monitoring events, identifying trends, and preparing for taking advantage of opportunities that are largely unpredictable in both their nature and timing.

The enablement and focus on the individual will undoubtedly also increase the level of societal stress and anxiety, as a significant number of persons find it difficult to keep pace with the tempo of technology driven change. Specifically, there is likely to be an increasing demand for freedom without a commensurate willingness to exercise self-constraint. At the same time the rapidly increasing desire for a higher quality of life and the mounting aspirations for personal achievement will for most persons fall short of their expectations.

Over the past two decades organizations have become increasingly aware of the potential value of their members. To harness and exploit these human assets, organizations have examined their structure. Efforts in this area are focused on creating an environment that encourages and facilitates the acquisition, sharing, and application of knowledge. Commonly referred to as *knowledge management*, these efforts have the goal of effectively developing and utilizing the human capital in an organization. More specifically, the objective of knowledge management is to enable all human and organizational capabilities and relationships for the benefit of the individual and the organization. This requires the encouragement of every member of the organization to be a contributor and a potential decision maker. How can this be achieved? Decentralization and concurrency are principal characteristics of knowledge management, aimed at creating an environment that builds relationships for the purpose of maximizing interaction, diversity, responsiveness, and flexibility.

In this respect knowledge management views an organization and its external environment as a complex adaptive system of many component parts acting in parallel. The principal component parts of the organization are the human players, including not only the employees but also the external individuals and groups that the organization interacts with. In this respect an organization may be characterized as many agents acting in parallel. Each agent is always ready to interact with the system, proactively and reactively responding to whatever the other agents are doing. As a network, such a complex and adaptive system is by its very nature highly decentralized. Any coherent behavioral patterns of the system are due to the collective competitive and cooperative activities of its parts. It follows that such a system has many levels of organization, with the agents at any level contributing in a building block manner to the agents at a higher level. For example, a group of individuals will form a team or department, a number of departments will form a division, and so on through an organization. Most importantly the organization is constantly changing, revising and rearranging these building blocks through its activities.

Two essential requirements for the relative success of an organization, within the context of such a dynamically adaptive environment, are *anticipation of the future* and *communication*. Neither of these are necessarily akin to human nature. The fundamental (i.e., biological) experience-based nature of the human cognitive system provides us with few tools to deal with situations that are not the same or at least similar to past experiences. Anticipation of the future therefore represents a precarious excursion into unknown territory that is typically accompanied by an elevated level of anxiety due to uncertainty, frustration and fear. The uncertainty stems from the unknown nature of the future, which differs fundamentally from the certainty of the past. Therefore from a human point of view, dealing with the future represents an emotional effort that challenges our confidence to survive and prosper within our environment. We become frustrated as we see many of the methods and tools that have allowed us to survive and prosper in the past, progressively fail as we try to apply them to new conditions and situations. We are forced to stumble along as we learn by trial and error. It is therefore only natural for us human beings to avoid any excursions into the future unless they are forced upon us. With few exceptions we tend to cling to the apparently safe domain of the past, unless we are compelled to face the present and future by developments in our environment that severely threaten the comfort level of our current role. Clearly, the requirement for anticipation in a successful organization is not naturally satisfied by its human players and must therefore be continuously fostered by other stimuli.

Since in this portrayal of an organization as a complex adaptive system the success of the organization depends greatly on the continuous interaction of its component parts, the maintenance of open communication channels between the human players of an organization is an essential requirement for knowledge management. The more active individuals or groups of players are the more critical the exchange of information and knowledge becomes to the welfare of the organization. Yet, there is a natural tendency for human beings to reduce their external interactions as they become more focused on their activities and, often to an even greater extent, as these activities appear to become successful. Both the concentration of their attention and the selfishness of their ambitions mitigate against the sharing of the knowledge acquired through their efforts. To overcome these natural human tendencies an organization is well advised to be sensitive to the individual needs and aspirations of its *enabled employees*, proactive in fostering

formal and informal communication channels, and above all valuing the contributions of its human resources.

The kind of computing infrastructure that has emerged to support the information management demands of a knowledge-based organizational structure is one that is readily accessible from any physical location, that is adept in exchanging information, and that shields the human user from the technical intricacies of its implementation. These characteristics are the foundational design principles of a Service-Oriented Architecture (SOA) computer software infrastructure. In a SOA-based information management environment, capabilities are provided in the form of software services that are decoupled from each other and today typically accessed through a Web interface.

The core of the infrastructure is a services management framework, which takes care of all of the data mappings and protocol changes that may be required to exchange data between services. However, all of these transformations are performed in a manner that is transparent to the individual services. In other words a service that requests some action by another service, such as the calculation of the interest due on a loan or the interpretation of a weather report, does not need to know what other service to send this request to or how to send the request. Similarly, the service that is selected by the infrastructure to process the request does not have to know which particular service this request originated from nor how to transmit the reply to the originator.

While the concept of a SOA-based information management environment is compellingly simple and appealing, the technical implementation of this concept is of course much more complicated and depends greatly on the availability of powerful computer hardware and communication facilities. In addition, there are issues related to data security and user privacy that still need to be resolved and may in some cases even require legislative action. Nevertheless, the convenience and flexibility provided by a services-based software architecture are likely to outweigh these concerns and persuade the users to accept some privacy risks.

Jens Pohl, June 2009

(jpohl@calpoly.edu) (www.cadrc.calpoly.edu)

InterSymp-2009
International Conference on Systems Research, Informatics and Cybernetics

Focus Symposium
on
Knowledge Management Systems

Tuesday, August 4, 2009

TABLE OF CONTENTS

"Intelligent Information Management Tools in a Service-Oriented Software Environment" **9**
***Jens G. Pohl**, PhD, Collaborative Agent Design Research Center, Cal Poly, San Luis Obispo, California, USA.*

"Modular Information Modeling: A Service Oriented Architecture for Distributed Data" **21**
***Sumantro Ray, Raj Mehta** and **Rao Anumolu,** ASR International Corporation, Hauppauge, New York, USA.*

"Open Standards-Based Building Information Modeling as the Foundation for Knowledge Management in the AEC Industry " **29**
***Igor Svetel,** PhD, Innovation Center, Faculty of Mechanical Engineering, Belgrade, Serbia and **Milica Pejanovi,** Faculty of Architecture, University of Belgrade, Serbia.*

"A Scheme and Formal Tools for Transforming the Existing Web into Semantic Web of a New Generation " **39**
***Vladimir Fomichov**, PhD, Department of Innovations and Business in the Sphere of Informational Technologies, Faculty of Business Informatics, State University –Higher School of Economics, Kirpichnaya str. 33, Moscow, Russia.*

"The Representation of Context in Computer Software" **51**
***Hisham Assal,** PhD**, Kym J. Pohl, and Jens G. Pohl,** PhD, Collaborative Agent Design Research Center, Cal Poly, San Luis Obispo, California, USA.*

"Conveyance Estimator Ontology: Conceptual and Object Models" **63**
***Xiaoshan Pan,** PhD, CDM Technologies Inc., San Luis Obispo, California, USA.*

“Increasing the Expressiveness of OWL Through Procedural Attachments” **75**
***Dennis Taylor**, CDM Technologies Inc., San Luis Obispo, California, USA.*

Intelligent Information Management Tools in a Service-Oriented Software Environment

Jens Pohl, Ph.D.
Executive Director, Collaborative Agent Design Research Center (CADRC)
California Polytechnic State University (Cal Poly)
San Luis Obispo, California, USA

Abstract

This paper draws attention to the increasing need for agile and adaptive software environments that are capable of supporting rapid re-planning during the execution of time-critical operations involving commercial end-to-end supply chain transaction sequences, as well as disaster response and military missions. It is argued that such environments are currently best served by information-centric software tools executing within a service-oriented paradigm. Service-oriented architecture (SOA) design concepts and principles are described, with a focus on the functions of the services management framework (SMF) and enterprise service bus (ESB) components. Differentiating between data-centric and information-centric services, it is suggested that only intelligent software services, particularly those that incorporate an internal representation of context in the form of an ontology and agents with reasoning capabilities, are able to effectively address the need for agile and adaptive planning, re-planning and decision-support tools.

The paper concludes with a description of the design components of a business process management (BPM) system operating within a SOA-based infrastructure, followed by a brief discussion of *Cloud* computing promises and potential user concerns.

Keywords: adaptive, agile, APEX, cloud computing, BPEL, business process execution language, BPM, business process management, choreographer, data-centric, enterprise service bus, ESB, information-centric, mediator, registry, services management framework, SMF, service-oriented architecture, SOA

Need for Adaptive Planning Tools

There is an increasing need in industry and government for planners and decision-makers to be able to rapidly re-plan during execution. Experience has shown that the best-laid plans will likely have to be changed during implementation. Operational environments are often impacted by events or combinations of factors that were either not foreseen during the planning stage or were thought to be unlikely to occur. In commerce, where *just in time inventories* have become an acknowledged cost-saving measure, suppliers and shippers are particularly vulnerable to the disruption of end-to-end supply chain sequences, such as inclement weather conditions, traffic congestion, accidents, equipment malfunction, and human error.

Military commanders, who often deal with extremely time-critical and human life endangering operations have learned from bitter experience that agile planning tools are essential for their ability to rapidly adapt to changing mission conditions. It can be argued that an information

management environment, with an agile planning capability of the type implied by the stated objectives of the *Adaptive Planning and Execution (APEX)*[1] process recently adopted by the U.S. military forces, requires both the ability to automatically interpret data in context and the flexibility to provide access to decision-support tools regardless of whether these are part of the same software application or another application.

This argument is based on the definition of *agility* as the ability to rapidly adapt to changing conditions, and has two implications. First, in a real world environment the operational data that enter a particular application may not adhere exactly to the specifications on which the design of the software was originally based. An agile software application will therefore need to have the ability to automatically interpret the incoming data within the appropriate context and make the necessary processing adjustments. Second, under such dynamic conditions it is likely that the user will have a need for tools that were not foreseen during the design of the application and are therefore not available. An agile software environment will therefore have to provide access to a wide range of tools, at least some of which may not be an integral component of the particular application that the operator is currently using. This suggests a system environment in which software tools can be seamlessly accessed across normal application domain boundaries. This is the objective of an information management environment that is based on the service-oriented concepts and principles described in this paper.

Information-Centric vs. Data-Centric

There are several reasons why computer software must increasingly incorporate more and more *intelligent* capabilities (Pohl 2005). Perhaps the most compelling of these reasons relates to the current data-processing bottleneck. Advancements in computer technology over the past several decades have made it possible to store vast amounts of data in electronic form. Based on past manual information handling practices and implicit acceptance of the principle that the interpretation of data into information and knowledge is the responsibility of the human operators of the computer-based data storage devices, emphasis was placed on storage efficiency rather than processing effectiveness. Typically, data file and database management methodologies focused on the storage, retrieval and manipulation of data transactions[2], rather than the *context* within which the collected data would later become useful in planning, monitoring, assessment, and decision-making tasks.

The term *information-centric* refers to the representation of information, as it is available to software modules, not to the way it is actually stored in a digital machine. This distinction

[1] Adaptive Planning and Execution Roadmap II, AO Review (Draft), Joint Chiefs of Staff, 8 February 2007.

[2] Most large organizations, including the Military, are currently forced to dedicate a significant portion of their operating budget, staff, project budgets, and time, on the piecemeal resolution of ad hoc problems and obstacles that are symptoms of an overloaded data-centric environment. Examples include: data bottlenecks and transmission delays resulting in aged data; temporary breakdown of data exchange interfaces; inability to quickly find critical data within a large distributed network of data-processing nodes; inability to interpret and analyze data within time constraints; and, determining the accuracy of the data that are readily available. This places the organization in a *reactive* mode, and forces it to expend many of its resources on solving the symptoms rather than the core problem. In contrast, an information-centric environment is capable of supporting: (1) the automatic filtering of data by placing data into an information context; (2) the automated reasoning of software agents as they monitor events and assist human planners and problem solvers in an intelligent collaborative decision-making environment; and, (3) autonomic computing capabilities.

between *representation* and *storage* is important, and relevant far beyond the realm of computers. When we write a note with a pencil on a sheet of paper, the content (i.e., meaning) of the note is unrelated to the storage device. A sheet of paper is designed to be a very efficient storage medium that can be easily stacked in sets of hundreds, filed in folders, folded, bound into volumes, and so on. As such, representation can exist at varying levels of abstraction. The lowest level of representation considered is *wrapped* data. Wrapped data consists of low-level data, for example a textual e-mail message that is placed inside some sort of an e-mail message object. While it could be argued that the e-mail message is thereby objectified it is clear that the only objectification resides in the shell that contains the data and not the e-mail content. The message is still in a data-centric form offering a limited opportunity for interpretation by software components.

A higher level of representation endeavors to describe aspects of a domain as collections of inter-related, constrained objects. This level of representation is commonly referred to as an information-centric ontology. At this level of representation context can begin to be captured and represented in a manner supportive of software-based reasoning. This level of representation (i.e., context) is an empowering design principle that allows software to undertake the interpretation of operational data changes within the context provided by the internal information model (i.e., ontology).

Even before the advent of the Internet and the widespread promulgation of SOA concepts it was considered good software design and engineering practice to build distributed software systems of loosely coupled modules that are able to collaborate by subscription to a shared information model. The principles and corresponding capabilities that enable these software modules to function as decoupled services include (Pohl 2007):

- An internal *information* model that provides a usable representation of the application domain in which the service is being offered. In other words, the *context* provided by the internal information model must be adequate for the software application (i.e., service) to perform as a useful adaptive set of tools in its area of expertise.
- The ability to *reason* about events within the context provided by the internal information model. These reasoning capabilities may extend beyond the ability to render application domain related services to the performance of self-monitoring maintenance and related operational efficiency tasks.
- Facilities that allow the service to *subscribe* to other internal services and understand the nature and capabilities of these resources based on its internal information model[3].
- The ability of a service to understand the notion of *intent* (i.e., goals and objectives) and undertake self-activated tasks to satisfy its intent. Within the current state-of-the-art this capability is largely limited by the degree of context that is provided by the internal information model.

[3] This must be considered a minimum system capability. The full implementation of a web services environment should include facilities that allow a service to *discover* other external services and understand the nature and capabilities of these external services.

Additional capabilities that are not yet able to be realized in production systems due to technical limitations, but have been demonstrated in the laboratory environment, include: the ability of a service to *learn* through the acquisition and merging of information fragments obtained from external sources with its own internal information model (i.e., dynamically *extensible* information models); extension of the internal information model to include the internal operational domain of the software application itself and the role of the service within the external environment; and, the ability of a service to increase its capabilities by either *generating* new tools (e.g., creating new agents or cloning existing agents) or automatically *searching* for external assistance.

Service-Oriented Architecture (SOA)

The notion of *service-oriented* is ubiquitous. Everywhere we see countless examples of tasks being performed by a combination of services, which are able to interoperate in a manner that results in the achievement of a desired objective. Typically, each of these services is not only *reusable* but also sufficiently *decoupled* from the final objective to be useful for the performance of several somewhat similar tasks that may lead to quite different results. For example, a common knife can be used in the kitchen for preparing vegetables, or for peeling an orange, or for physical combat, or as a makeshift screwdriver. In each case the service provided by the knife is only one of the services that are required to complete the task. Clearly, the ability to design and implement a complex process through the application of many specialized services in a particular sequence has been responsible for most of mankind's achievements in the physical world. The key to the success of this approach is the *interface*, which allows each service to be utilized in a manner that ensures that the end-product of one service becomes the starting point of another service.

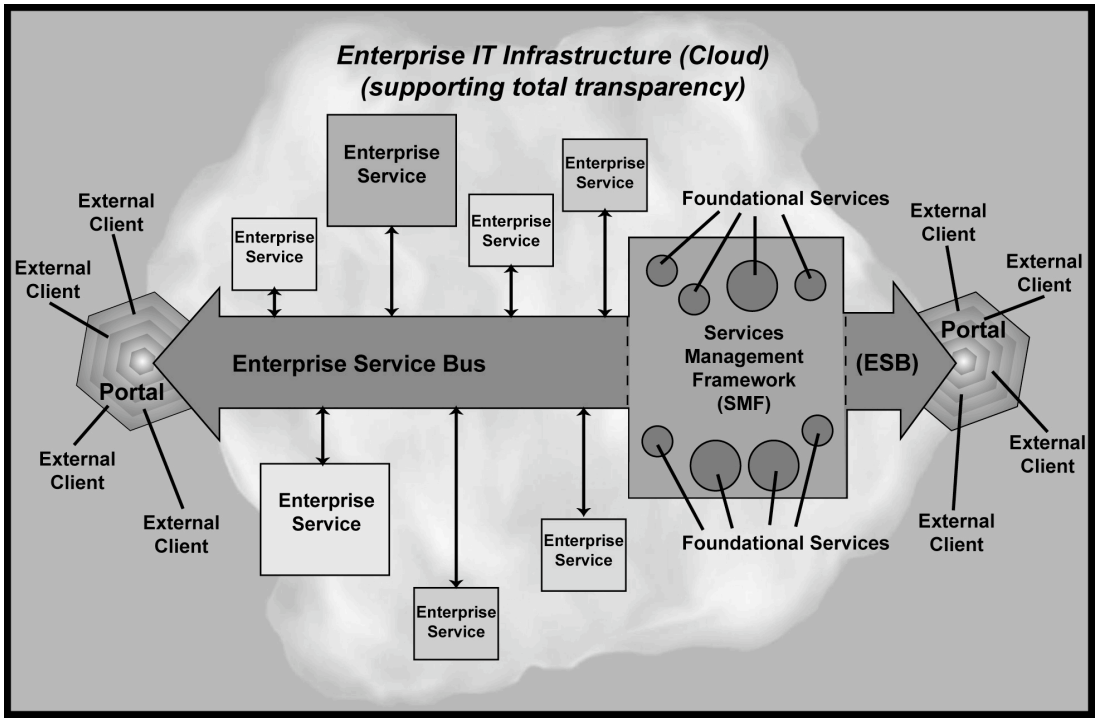


Figure 1: Principal components of a conceptual SOA implementation

In the software domain these same concepts have gradually led to the adoption of Service-Oriented Architecture (SOA) principles. While SOA is by no means a new concept in the software industry it was not until Web services became available that these concepts could be readily implemented (Erl 2005). In the broadest sense SOA is a software framework for computational resources to provide services to customers, such as other services or users. The Organization for the Advancement of Structured Information (OASIS)[4] defines SOA as a *"… paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains"* and *"…provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects with measurable preconditions and expectations"*. This definition underscores the fundamental intent that is embodied in the SOA paradigm, namely *flexibility*. To be as flexible as possible a SOA environment is highly modular, platform independent, compliant with standards, and incorporates mechanisms for identifying, categorizing, provisioning, delivering, and monitoring services.

The principal components of a conceptual SOA implementation scheme (Figure 1) include a Services Management Framework (SMF), various kinds of foundational services that allow the SMF to perform its management functions, one or more portals to external clients, and the enterprise services that facilitate the ability of the user community to perform its operational tasks.

Services Management Framework (SMF): A Services Management Framework (SMF) is essentially a SOA-based software infrastructure that utilizes tools to manage the exchange of messages among enterprise services. The messages may contain requests for services, data, the results of services performed, or any combination of these. The tools are often referred to as foundational services because they are vital to the ability of the SMF to perform its management functions, even though they are largely hidden from the user community. The SMF must be capable of:

- Undertaking any transformation, orchestration, coordination, and security actions necessary for the effective exchange of the message.
- Maintaining a loosely coupled environment in which neither the service requesters nor the service providers need to communicate directly with each other; - or even have knowledge of each other.

A SMF may accomplish some of its functions through an Enterprise Service Bus (ESB), or it may be implemented entirely as an ESB.

Enterprise Service Bus (ESB): The concept of an Enterprise Service Bus (ESB) greatly facilitates a SOA implementation by providing specifications for the coherent management of services. The ESB provides the communication bridge that manages the exchange of messages among services, although the services do not necessarily know anything about each other. According to Erl (2005) ESB specifications typically define the following kinds of message management capabilities:

[4] OASIS is an international organization that produces standards. It was formed in 1993 under the name of SGML Open and changed its name to OASIS in 1998 in response to the changing focus from SGML (Standard Generalized Markup Language) to XML (Extensible Markup Language) related standards.

- *Routing:* The ability to channel a service request to a particular service provider based on some routing criteria (e.g., static or deterministic, content-based, policy-based, rule-based).
- *Protocol Transformation:* The ability to seamlessly transform the sender's message protocol to the receiver's message protocol.
- *Message Transformation:* The ability to convert the structure and format of a message to match the requirements of the receiver.
- *Message Enhancement:* The ability to modify or add to a sender's message to match the content expectations of the receiver.
- *Service Mapping:* The ability to translate a logical business service request into the corresponding physical implementation by providing the location and binding information of the service provider.
- *Message Processing:* The ability to accept a service request and ensure delivery of either the message of a service provider or an error message back to the sender. Requires a queuing capability to prevent the loss of messages.
- *Process Choreography and Orchestration:* The ability to manage multiple services to coordinate a single business service request (i.e., choreograph), including the implementation (i.e., orchestrate). An ESB may utilize a Business Process Execution Language (BPEL) to facilitate the choreographing.
- *Transaction Management:* The ability to manage a service request that involves multiple service providers, so that each service provider can process its portion of the request without regard to the other parts of the request.
- *Access Control and Security:* The ability to provide some level of access control to protect enterprise services from unauthorized messages.

There are quite a number of commercial off-the-shelf (COTS) ESB implementations that satisfy these specifications to varying degrees A full ESB implementation would include four distinct components (Figure 2): Mediator; Service Registry; Choreographer; and, Rules Engine. The Mediator serves as the entry point for all messages and has by far the largest number of message management responsibilities. It is responsible for routing, communication, message transformation, message enhancement, protocol transformation, message processing, error handling, service orchestration, transaction management, and access control (security).

The Service Registry provides the service mapping information (i.e., the location and binding of each service) to the Mediator. The Choreographer is responsible for the coordination of complex business processes that require the participation of multiple service providers. In some ESB implementations the Choreographer may also serve as an entry point to the ESB. In that case it assumes the additional responsibilities of message processing, transaction management, and access control (security). The Rules Engine provides the logic that is required for the routing, transformation and enhancement of messages. Clearly, the presence of such an engine in combination with an inferencing capability provides a great deal of scope for adding higher levels of intelligence to an ESB implementation.

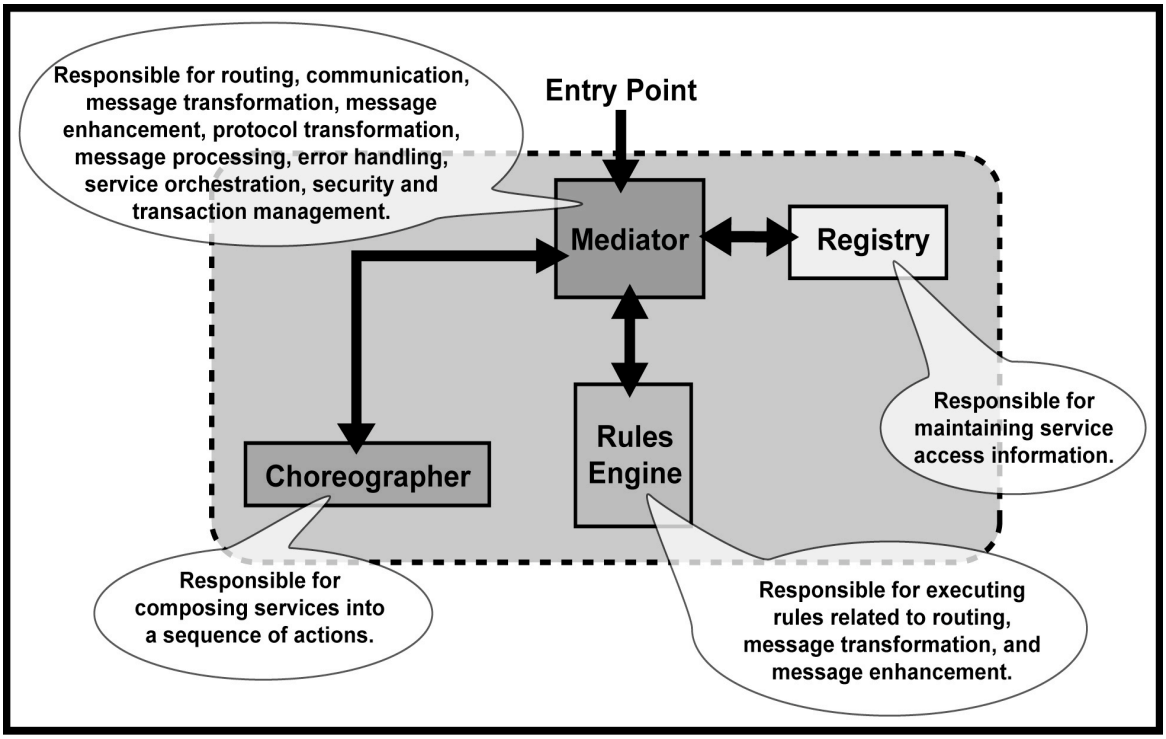


Figure 2: Primary ESB components

Typical Service Requester and Service Provider Scenario

The following sequence of conceptual steps that must be taken by the SMF to support a SOA system environment is not inclusive of every variance that might occur. It is intended to provide a brief description of the principal interactions involved (Figure 3).

While the Service Requester knows that the Mediator is the entry point of the ESB component of the SMF and what bindings (i.e., protocols) are supported by the Mediator, it does not know which Service Provider will satisfy the request because it knows nothing about any of the other enterprise services that are accessible through the Mediator. Therefore, the conceptual SOA-based infrastructure shown in Figure 1 is often referred to as a *Cloud*.

The Mediator is clearly in control and calls upon the other primary components of the ESB if and when it requires their services. It requests the *handle* (i.e., location and mappings) of the potential Service Providers from the Service Registry. If there are multiple Service Provider candidates then it will have to select one of these in Step (6) to provide the requested service. The Mediator will invoke any of the foundational services in the SMF to validate (i.e., access control), translate, transform, enhance, and route the message to the selected Service Provider. The latter is able to accept the message because it is now in a data exchange format that the Service Provider supports.

Similar transformation and mapping actions are taken by the Mediator after it receives the reply message from the Service Provider, so that it complies with the data exchange format supported by the Service Requester. On receiving the response message the Service Requester does not

know which service responded to the request, nor did it have to deal with any of the data exchange requirements of the Service Provider.

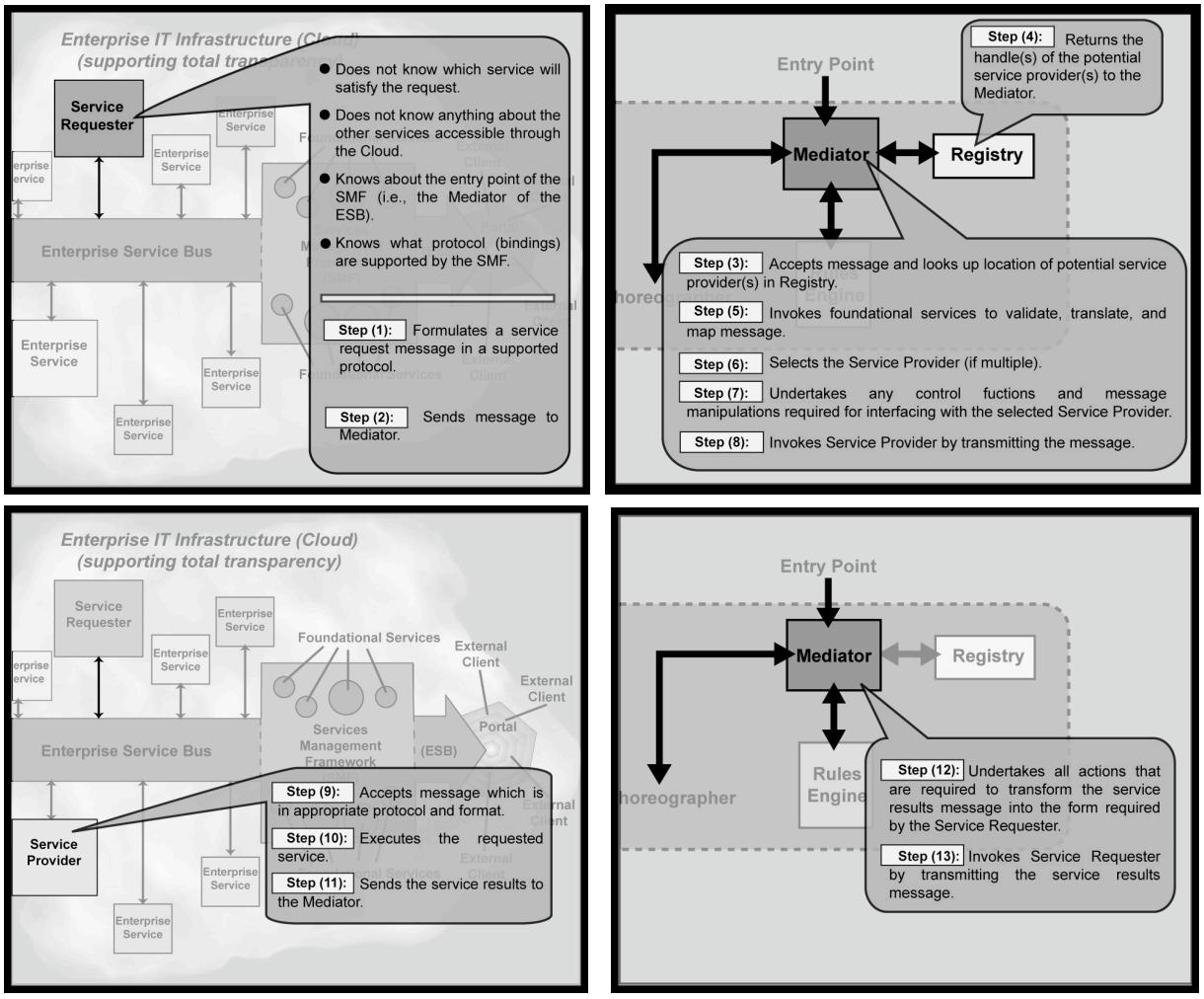


Figure 3: Conceptual *Cloud* operations

Business Process Management (BPM)

From a general point of view, Business Process management (BPM) is the orchestration of activities between people and systems. More specifically, BPM is a method for actively defining, executing, monitoring, analyzing, and subsequently refining manual or automated business processes. In other words, a business process is essentially a sequence of related, structured activities (i.e., a workflow) that is intended to achieve an objective. Such workflows can include interactions between human users, software applications or services, or a combination of both.

In a SOA-based information management environment this orchestration is most commonly performed by the Choreographer component of the ESB (Figure 2). Based on SOA principles, a sound BPM design will decompose a complex business process into smaller, more manageable elements that comply with common standards and reuse existing solutions.

The BPM design solution should be based on an analysis of the problem within both its local and global contexts (Figure 4). It must describe and support the local business process requirements as its primary objective and yet seamlessly integrate this micro perspective into a global view. Successful integration of these two perspectives will require an understanding of external interactions and the compliance parameters that apply to interprocess protocols. The principal components of a BPM design solution include a Business Process Execution Language (BPEL) engine, a graphical modeling tool, business user and system administration interfaces, internal and external system interactions, and persistence (Figure 5).

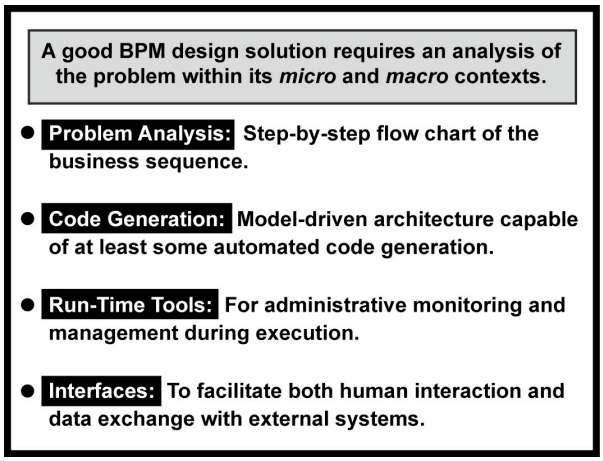


Figure 4: BPM design requirements

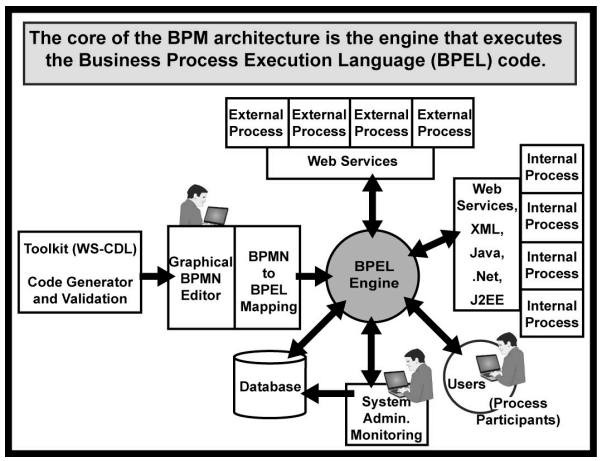


Figure 5: BPM design components

BPEL Engine: BPEL, which is the preferred process language, is normally XML-based[5] and event driven. The BPEL Engine is responsible for detecting events, executing the appropriate next step in the business process sequence, and managing outbound message calls.

Graphical Editor: Effective communication during design is greatly facilitated by a standard system of notation that is known to all parties involved in the design process, and a graphical tool that allows design solutions to be represented in the form of diagrams. Both the Business Process Modeling Notation (BPMN)[6] and the Unified Modeling Language (UML)[7] Activity Diagram provide the necessary capabilities. However, BPMN is normally preferred because it incorporates BPEL mapping capabilities and is considered to be the more expressive notation. Whichever graphical modeling tool is chosen it should be capable of representing the different views of the process that are desired by the business user and the technical user. The business user is

[5] The Extensible Markup Language (XML) is a general purpose specification that allows the content of a document to be defined separately from the formatting of the document.

[6] BPMN provides a graphical representation for describing a business process in the form of a workflow diagram. It was developed by the Business Process Management Initiative (BPMI) and is now maintained by the Object Management Group following the merging of these two organizations in 2005.

[7] The Unified Modeling Language (UML) provides a standard notation for modeling systems and context based on object-oriented concepts and principles (Booch G., J. Rumbaugh and I. Jacobson (1999); 'The Unified Modeling Language User Guide'; Addison-Wesley, New York, New York.)

interested in the overall flow of the process, while the technical user is interested in the more detailed behavioral characteristics of each step.

User-interfaces: Typically, separate user-interfaces are required for the business user who has a functional role in the business process and may from time to time be required to interact with the BPEL Engine, and the system administrator who may be monitoring the task flow for reactive or proactive system maintenance reasons. The business users essentially require a *worklist*[8] interface that allows them to contribute manual tasks to the automated BPM process. This should be a user-friendly, role-based interface with process status reports and error correction capabilities. The system administrators require a user-interface that allows them to perform a host of management tasks including: defining a process (i.e., find, activate, deactivate, remove, or add); controlling the execution of processes known to the BPEL Engine and worklist tasks or activities (i.e., find, suspend, resume, or terminate); managing user roles (i.e., add, modify, or remove users and roles from applications); and, configuring application connections.

Both the business and system administration user-interfaces must incorporate security measures to prevent unauthorized access and ensure that only authorized role-based actions can be executed.

System interactions: A business process is likely to involve both internal and external system interactions. In general terms these interactions may be characterized as four distinct modes: process receives a message from another system; process receives a message and sends a response; process sends a message to another system; and, process sends a message and waits for a response. External interactions are typically choreographed as web services, with a wide variety of system interfaces being supported through a generic adapter facility. This means that the BPEL Engine must include a web services listener capable of accepting an inbound message (e.g., in SOAP[9] format), insert it into the runtime engine, obtain a response (if any), and send out the response as a SOAP message. Internal interactions are typically either client-server interfaces to other systems executing on the enterprise network or inline code snippets.

Persistence: To survive the inevitable need to restart the BPEL Engine the current process state must be stored in a database. Tables in the database typically include: process definition; process execution state; message content and identification code; process variables; activity execution state; and, worklist task execution state.

While BPM and SOA concepts are closely connected, they are certainly not synonymous. Described more precisely, a SOA-based system environment provides the enabling infrastructure for BPM by separating the functional execution of the business process from its technical implementation.

---

[8] A BPM *worklist* allows a manual task to be assigned to a user and track the progress of that task. In this way the human user can be the source of events that trigger the BPEL Engine.

[9] The Simple Object Access Protocol (SOAP) is a protocol specification for the exchange of data among web services. It utilizes XML as its message format and depends on other protocols, such as Remote Procedure Call (RPC) and Hypertext Transfer Protocol (HTTP) for transmitting the message.

In Conclusion: Cloud Computing

The concept of *Cloud* computing as a massively scalable, user-transparent computing resource that can be readily accessed by multiple users across a global network is indeed a compellingly attractive proposition. Combined with the SOA design and implementation principles described above, the *Cloud* not only takes care of all of the intricate technical interoperability and data exchange incompatibility issues that have plagued computer users in the past, but also provides essentially ubiquitous access to powerful and seamlessly integrated computer-based capabilities as services. Naturally, multiple *Clouds* can be linked in a manner that is quite similar to the way services are registered within a particular *Cloud*. In such an environment neither the service requester nor the service provider needs to know, or even care, where the request originated and where it was processed, even if the request for services had to traverse several *Clouds* before the necessary service provider could be found.

It is of interest to note that this view of computing as a service is not new. During the 1960s and 1970s time-share computer systems, which linked multiple remote user terminals through modems to a central computing facility, provided a similar computing service. However, there were some major differences. First, access and data exchange was strictly confined to a single computer center and in most cases to the particular application that the user was authorized to use. Second, very little of the underlying computing environment was transparent to the user. Third, the users were almost as rigidly tied to their access terminal location as the service provider was tied to the location of its computer center. The time-share concept became obsolete as soon as the advent of microcomputers brought the computing power to the user.

We might ask: *Was it a desire by the computer users to have complete control over their computing resources or convenience that led to the preference of ownership over service?* While *Cloud* computing promises to overcome the inconvenience, immobility, and lack of interoperability constraints of the time-share service environment, it does pose other problems that will need to be overcome. Chief among these is the issue of data security. Will organizations be willing to entrust their proprietary data to a remote *Cloud* environment over which, in reality, they have little control? They must trust the *Cloud* service provider to not only maintain adequate internal security, but to resist even the most sophisticated and continuously changing external intrusion attempts. Also, as Robert Lucky (2009) recently wrote "… once all your petabytes of data are out there in the *Cloud*, can you ever get them back?

Finally, there is the question of user autonomy and control. Are current and will future privacy laws be sufficient to protect the user from a plethora of potential consumer abuses, for example, the automated collection of data about a user's activities in the *Cloud* without the need to actually trespass the data repositories themselves. Such data are already being collected by Internet service providers and utilized to determine collective and individual preferences for advertising and directed marketing purposes. Perhaps users will not be greatly concerned about the potential privacy infringements of such activities, and in the end the convenience and inexpensiveness of *Cloud* computing may become the deciding factors.

References

Burlton R. (2001); 'Business Process Management: Profiling from Process'; SAMS, Indianapolis, Indiana.

Chang J. (2005); 'Business Process Management Systems'; Auerbach Publications, Auerbach/Vogtland, Germany.

Erl T. (2005); 'Service-Oriented Architecture (SOA): Concepts, Technology, and Design'; Prentice Hall Service-Oriented Computing Series, Prentice Hall, Englewood Cliffs, New Jersey.

Havey M. (2005); 'Essential Business Process Modeling'; O'Reilly, Sebastopol, California.

Jeston J. and J. Nelis (2006); 'Business Process Management: Practical Guidelines to Successful Implementations'; Butterworth Hein Elsevier, United Kingdom.

Lucky R. (2009); 'Cloud Computing'; (under *Reflections*) IEEE Spectrum, Institute of Electrical and Electronics Engineers, 46(5), May (pp. 27).

Pohl J. (2005); 'Intelligent Software Systems in Historical Context'; in Jain L. and G. Wren (eds.); 'Decision Support Systems in Agent-Based Intelligent Environments'; Knowledge-Based Intelligent Engineering Systems Series, Advanced Knowledge International (AKI), Sydney, Australia.

Pohl J. (2007); 'Knowledge Management Enterprise Services (KMES): Concepts and Implementation Principles'; InterSymp-2007, Proceedings Focus Symposium on Representation of Context in Software, Baden-Baden July 31, Germany.

Taylor D. and H. Assal (2008); 'Using BPM as an Interoperability Platform'; C2 Journal, Special Issue on Modeling and Simulation, CCRP, Washington, DC, Fall.

Modular Information Modeling
A Service Oriented Architecture for Distributed Data

Sumantro Ray[1], **Raj Mehta**[2], **Rao Anumolu**

ASR International Corporation[3]
Hauppauge, NY, USA

Abstract

In today's world, the primary bottleneck for data management is not the collection of data, but its organization, processing, and systematic dissemination. Workforces are distributed and diverse, and data comes in various forms, from emails to documents to web-based forms. In such a data-rich world, systems that integrate all the different streams into a cohesive and coherent message are in great demand. A meaningful transformation of raw data into information, and subsequently into knowledge is helpful not only for understanding the patterns in the tapestry of such seemingly irreconcilable data, but also can lead to testable predictions and serendipitous steps in decision-making.

To that end, we have developed a web-based knowledge management system that integrates all the data at our organization, past and present, into one modular system that crunches this data from our diverse workforce. This system integrates data visualization, email analyses, audio read-outs, OCR, document indexing and search, and resource allocations and utilization (personnel, time) to generate a snapshot of the organization's output over any period. In addition, the system generates reports and email alerts, and helps streamline the organizational processes. In this paper, we present this generic architecture that can be a model for any organization that strives to capture data and generate information regarding its processes and activities.

This paper is an extension to our InterSymp-2007 paper. In that paper, we introduced the **collect-process-consume** model that effectively shields the complexity of the data-crunching and analysis from the end user. Over time, the data accumulates into a knowledge base, which the managers/consumers of the data can then refer to for institutionalized wisdom. In this paper, we extend this concept to a more rigorous SOA where different modules can easily be integrated in one coherent framework. Module interfaces are more tightly defined, and more rigorous version control is enforced. This leads to faster and parallelized development, tighter development cycles, minimized errors, higher efficiency, and uniform interfaces across our many projects.

Keywords

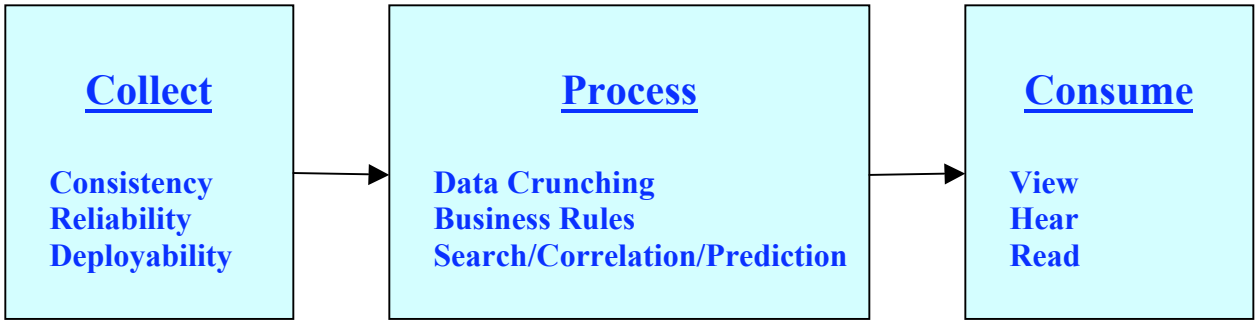
Distributed Systems, Database Systems, Visualization, Analysis, Correlation, Prediction, Search Engine, Data Extraction, Security, Service-Oriented-Architecture (SOA)

---

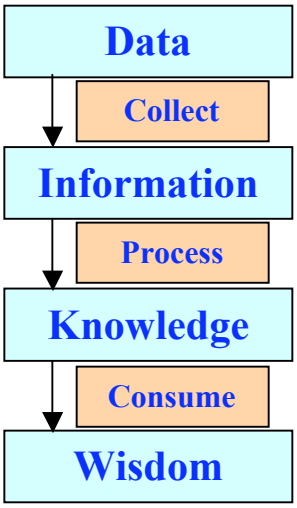
[1] sumantror@asrintl.com
[2] mehta@asrintl.com
[3] http://www.asrintl.com

Introduction

The **Collect-Process-Consume** models streamline data collection, processing, and consumption. This is a sufficiently high-level view of the data-flow of a project management framework. In any such framework, there will be more than one simultaneous producers and consumers of data. The goal of a robust system therefore is to make sure that this massively parallelized operation does not result in inconsistencies and/or race conditions (two entities waiting for the same resource) either in file system or the database.



- **Collection**
  - Streamline data collection
  - Reduce human error
  - Organize structured data
- **Processing**
  - Crunch data into information
  - Identify patterns and trends
  - Correlate unrelated processes
- **Consumption**
  - View: graphs, charts, figures
  - Hear: audio alerts and summaries
  - Read: email reports, text-messages



The CPC model is part of a larger philosophical construct. **Data**, when collected and stored in a structured fashion, becomes Information. **Information**, when processed and analyzed, leads to Knowledge. **Knowledge**, when consumed and acted upon, contribute to the collective **Wisdom**.

Our previous paper (InterSymp 2007) explains the CPC model in more detail. In this paper, we will focus on the modularization of the architecture. In such a framework, the system is implemented on a reasonably elaborate hardware/software environment, and consists of distinct modules, well-defined both in their functionality and design, and interfacing with each other. Such modules conform to the concept of the Service-Oriented-Architecture (SOA). Additionally, the design, development and testing of the programs also follow the SOA, thus leading to an efficient use of resources (man-power and time).

ASR Technology Platform

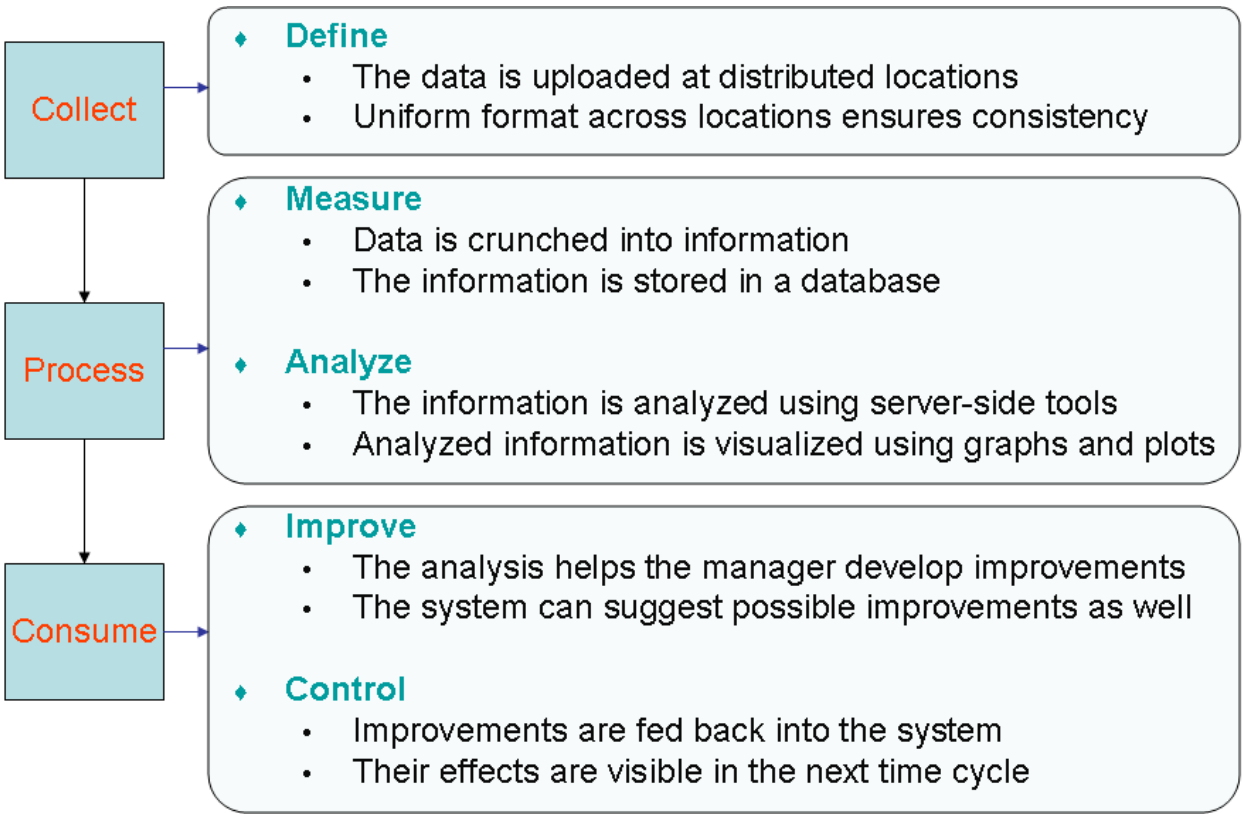
ASR Technology Platform is an integrated internet-based framework that provides dedicated services to clients. Following are some of the goals of this framework.

- Manage distributed work force
- Manage distributed work products
- Monitor customer deliverables as per Customer Business Rules
- Provide performance and productivity measures
- Capture field employees' knowledge
- Provide consistent training
- Make available current relevant documents / procedures / processes to field employees
- Collect field data and extract information using integrated tools:

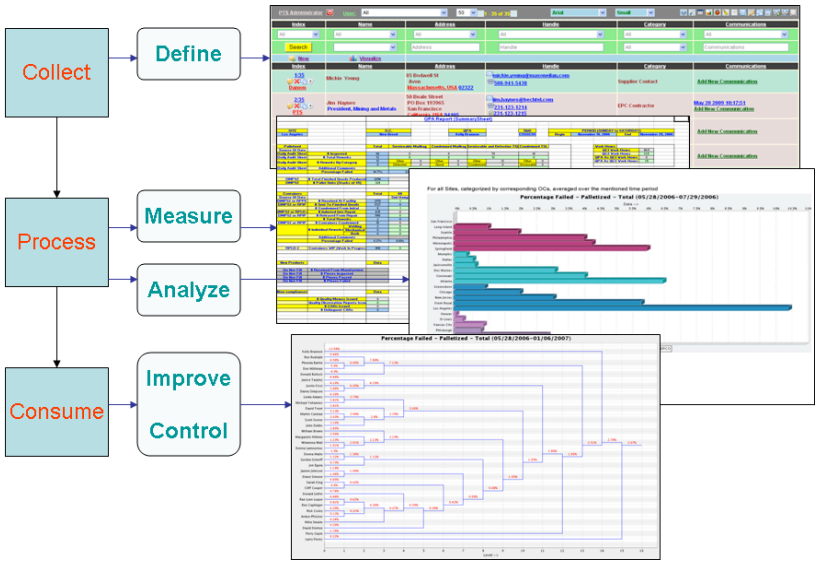
    - Data Analysis
    - Visualization
    - Dash-board
    - Search Engine

- Provide information to Customer

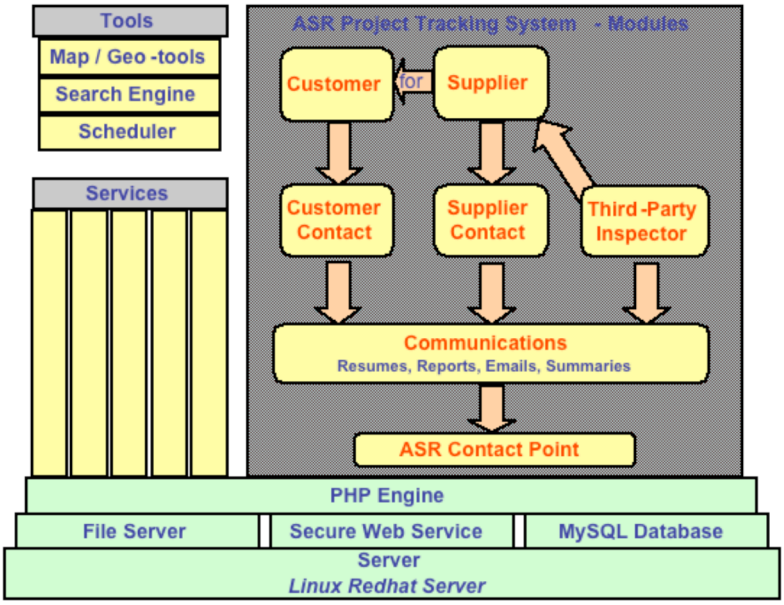
To that end, the CPC model can be further explained using the following chart. As is obvious, this model is not specific to implementation of a project management system. The framework is generic enough to allow implementations of other data management systems as well.



The following charts highlight a specific implementation of this framework, for one of our clients. Data is collected through structured Excel sheets. Server-side, the data is processed using business rules and graphs and charts. Users log in to a secure network to consume information, which include analyses, visualizations, forecasts and trends.



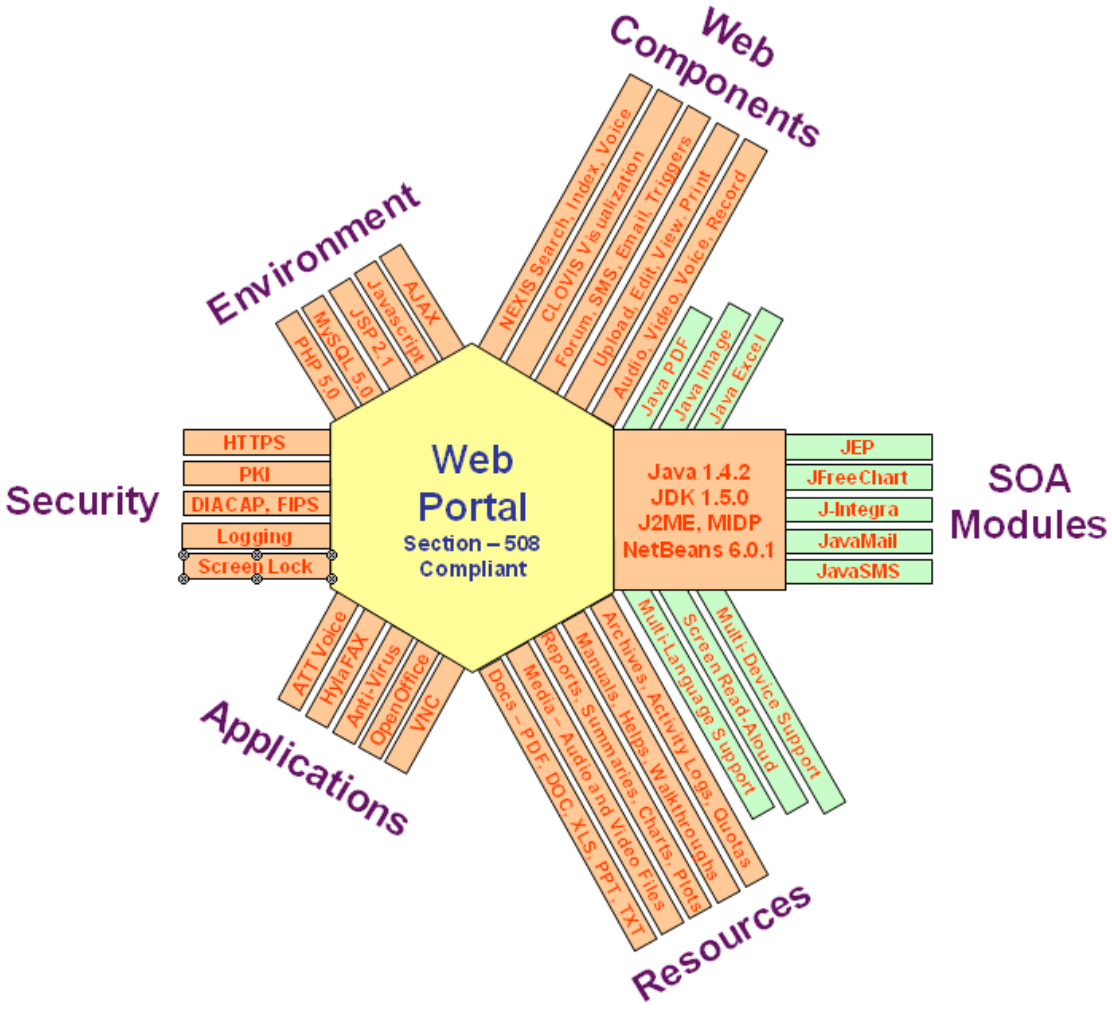
The Project Tracking System, or PTS, is the term used for the overall framework that implements this CPC architecture. ASR PTS sits on top of web, database and file servers. The *Services* are independent programs running on the server. The *Tools* are integrated modules in the website, offering various services for the customer and ASR contact point. A typical example could be a system to manage suppliers and inspectors using such a framework. Tools and Services are loosely independent SOA modules that integrate with any such project.



ASR SOA Framework

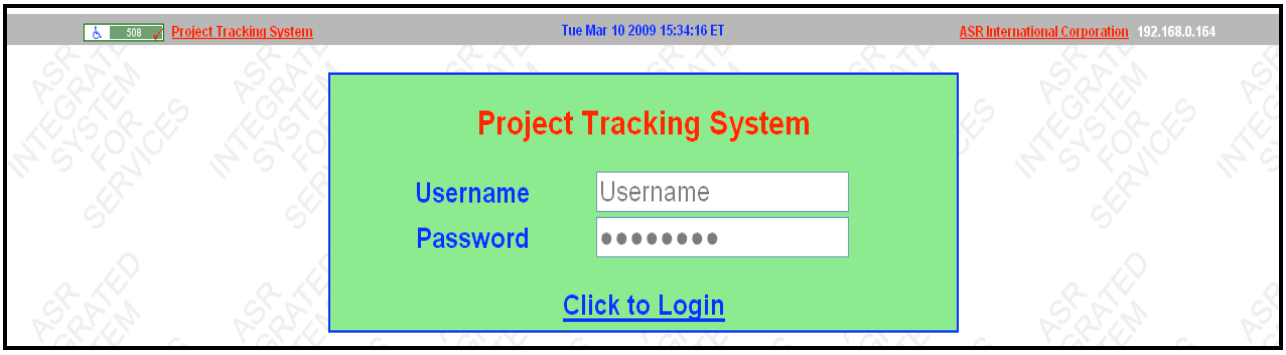
ASR hosts a web-based project management system (PTS) running behind a secure proxy server, encrypted with 128-bit Verisign SSL certificate. The PTS exists on a RedHat Linux driven quad-processor server, which runs MySQL database service and an apache web server. The server runs services such as text-to-voice, send-mail and OCR, and ASR proprietary modules such as an indexer/search engine, report generation engines, visualization engine, and email/SMS alerts.

The following graphic demonstrates ASR's capabilities in the context of the PTS. The PTS front-end is scripted using PHP and Javascript, while the backend proprietary tools are mostly written in Java. This ensures both security of the content, and speed at which it is accessed. Additional increased interactivity is guaranteed by using technologies such as AJAX.

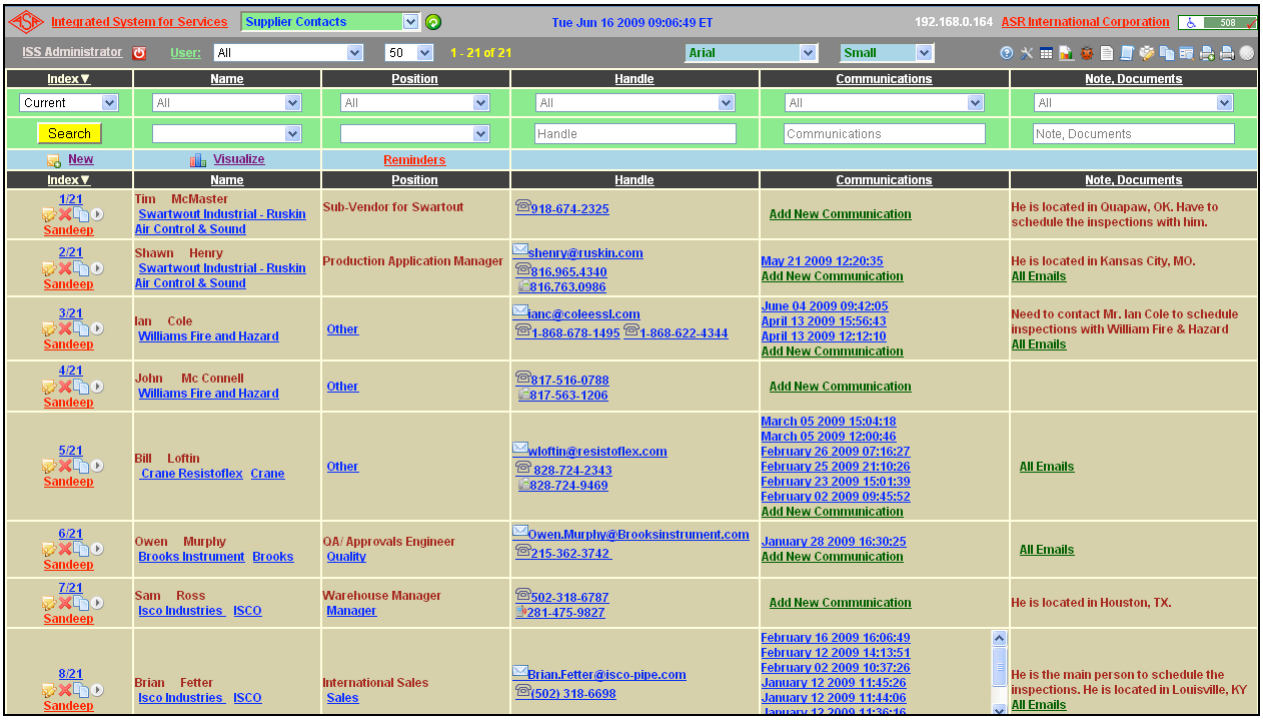


Each element in the hierarchy is self-contained, thus making the overall system modular, upgradeable and easily configurable. The modules correspond to different inter-connected datasets that contain data corresponding to different aspects of the Project Tracking System. The website integrates all these modules in a single, unified framework that is well-suited for supplier inspection, reporting, and inspector scheduling and management.

ASR PTS website and data are secure behind a password protected interface. This interface accepts user-name and password (DIACAP compatible) for local (with ASR corporate headquarters, for ASR contact points) and remote (for field inspectors, suppliers and customers) access.



Once logged in, a user is presented with a screen similar to the graphic below. For example, in the graphic below, the ASR contact point is navigating suppliers for a particular customer. Clicking on the controls at the top of the page will take users to the corresponding data-view.



The above graphic lists the suppliers, their addresses, corresponding customer and inspectors. Thus, this view gives a snapshot of all suppliers for this particular customer. Clicking on the name for any inspector will take the user to the view corresponding to that user, where details on the inspector's activities at various suppliers are presented. For a customer login, this view will be restricted only to suppliers supplying to that customer. Similarly, different filters will restrict access of the data to various suppliers, inspectors and customers.

Conclusion

As sites across the globe generate/upload data, the goal is to compress this data into manageable chunks of information that can be used by managers to help enable them take decisions that affect the day-to-day and long-term performance of the enterprise. The information thus generated can take the form of summary documents (which combine the data from various sites and present them as an aggregate), visualizations (charts, time-plots, histograms, trends and projections), audio (voice read-outs, voice-messages), and SMS (text notifications, *on-demand* information).

The SOA-based structure presented in this paper is robust, generalized framework based on the CPC (Collect-Process-Consume) model. The design, development, and testing of the system also adheres to this same philosophical construct, which leads to increased efficiency, faster turnaround time, and better designed interfaces. As systems become more and more complicated, such a system allows easy escalation of hardware, software and services, whilst also maintaining the ease of development and maintenance. The modular architecture allows plug-and-play use of tools such as the search engine, voice support, translation, mapping services, and communications.

Future Directions

We plan to make the system more visual in nature. As systems grow more complex, the push is towards visual consumption of information, as, we all know, a picture is worth a thousand words. The trick is to be able to generate aesthetically pleasing data-rich imagery that provides the information as fast as possible. In addition, more integration of AJAX-based tools will enhance interactivity. AI-based tools will allow forecasting and trend analyses, while learning-based modules will analyze user inputs and patterns to sort, search, and present data in a more user-friendly way.

References

A Distributed System for Clustered Visualization: A New Paradigm for Information Modeling, Sumantro Ray, et al, Intersymp 2007, 19th International Conference on Systems Research, Informatics and Cybernetics, Baden Baden, Germany

Open Standards-Based Building Information Modeling as the Foundation for Knowledge Management in the AEC industry

Igor Svetel

Innovation Center, Faculty of Mechanical Engineering, Belgrade, Serbia

Milica Pejanovi_

Faculty of Architecture, University of Belgrade, Serbia

Abstract

The Building Information Modeling (BIM) concept means to build a facility virtually, prior to building it physically, in order to work out problems and simulate and analyze potential impacts. This approach is supported by the development of a few open standards that provide lifecycle modeling of the facility and foster interoperability among all actors in the AEC industry both on syntactic and semantic levels. These standards enable a highly structured representation of information about buildings, but do not consider knowledge management problems. The Semantic Web initiative uses the same open standards approach and compatible technological background as BIM making it a perfect candidate for providing a knowledge management extension to BIM.

Keywords

Building Information Modeling, BIM, Industry Foundation Classes, IFC, knowledge management, ontology

Introduction

Recently a new wave of technical advances has swept the architecture, engineering and construction (AEC) industry. The term Building Information Modeling (BIM) is in use to denote a spectrum of both technological and methodological changes in the way the AEC profession is conducted. The objective of promoting BIM generated models as the international standard that will replace traditional AEC project documents spurred worldwide interest in the necessary adaptations of national AEC standards. The project 'Pilot Project: Application of the IFC (ISO/PAS 16739:2005) Standard in Drywall Systems' was commenced as the mean to achieve a

first experience with BIM technologies for the AEC industry in Serbia. The project goal is to create a BIM model based on the data from an actual building project and to evaluate the model by simulating events that have occurred in the actual design and construction processes. The process of creating a BIM model identified many errors in both the project data and the execution process on the construction site (Fig. 1). The first idea was to document those situations in a traditional written manual as a way of showing how BIM methods help to avoid errors. Additionally it was decided to represent them as the computer-based knowledge to further emphasize BIM method benefits. This paper presents the results of the process of analyzing current knowledge representations that will be used in our project and demonstrate how the open standard initiative enables an increasing level of interoperability. The research was conducted as a feasibility study on the availability of technologies that can be used to document differences between a model based on design data and a model based on data from the erected building.

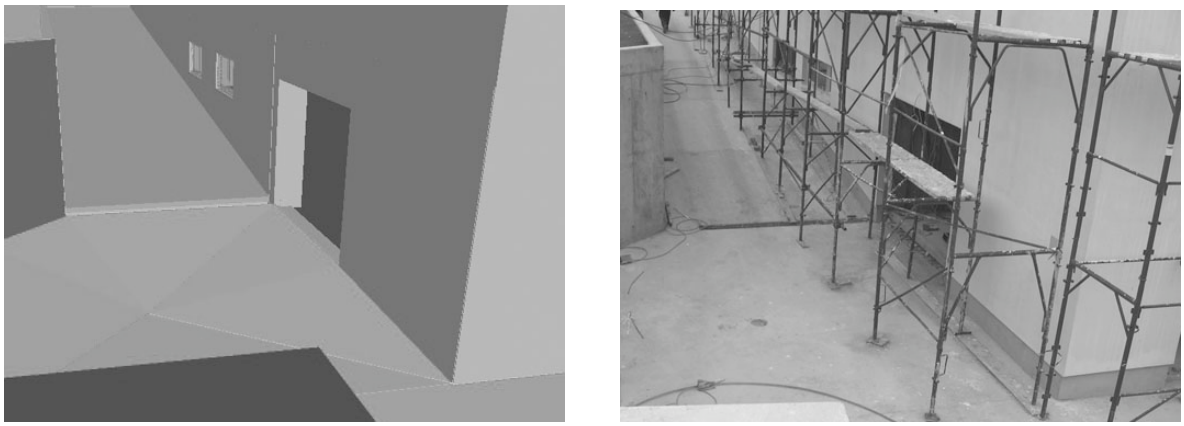


Figure 1. An example of an error detected on the site using a BIM model. The original project data produces a flat entrance to the garage and in the finished building a step appears at that location.

Open standards for AEC

Based on years of research on a general data model for AEC (Eastman et al. 1991; Björk 1989) the term Building Information Modeling (BIM) denotes a process of using Information Technology (IT) to model and manage data encompassing the whole facility lifecycle (Lee et al. 2006). The BIM concept means to build a facility virtually, prior to building it physically, in order to work out problems and simulate and analyze potential impacts. It is easier to fix a problem by moving element with a mouse than to demolish and rebuild elements on a construction site. The commercially developed BIM applications support creation of the computer-based facility model using parametric three-dimensional (3D) components with attached descriptive parameters that are necessary to fully identify particular elements. Still, those applications typically use proprietary data formats to represent facility models thus keeping all information locked in distinct software.

The need to establish interoperability among applications dealing with different phases of the facility lifecycle, such as architectural design, civil engineering, HVAC design, building construction, and facility management (FM), was met with the development of the Industry Foundation Classes (IFC) standard (Liebich, et al. 2007). The currently available model is Version 2, Revision 3 and is also registered as the ISO/PAS 16739:2005 standard. IFC is a neutral and open model whose development is conducted by the International Alliance for Interoperability (AIA), which has 550 member organizations in 24 countries. The standard provides the following basic functionality:

- Data interchange without information loss among all AEC and facility management (FM) applications.
- Unified model-based description of all building components.
- Information on the graphical representation of components.
- Description of relationships with other components and their location in the whole structure.
- Link to property and classification data, and access to external libraries.

The open specification of the IFC data model allows commercial software developers to write interfaces for their software that enable exchange and sharing of the same data in the same format with other software applications, regardless of the internal data structure of any individual software application. All leading software companies like Autodesk, Bentley System, Graphisoft, Nemetschek, Data Design System, Solibri, Tekla, Archimen Group, Vector–Works, etc. support IFC in their applications.

Being an object-oriented data model, the IFC standard is comprised of class definitions representing all things and events occurring in the facility lifecycle. At the top of the hierarchy is a domain layer that describes classes related to basic functional units: building controls, plumbing and fire protection, structural elements, structural analysis, heating, ventilation and air conditioning, electrical circuits, architecture, construction and facilities management. Below that layer rests the interoperability layer that defines all classes essential for connection and cooperation among disciplines. Next is the core layer, containing basic model classes depicting controls, products, and processes. The resource layer is at the bottom, embodying classes that represent all building elements. Elements encompass not only physical components, as traditional models, but also actors and their roles, time, price, approval, etc.

The IFC standard does not produce one monolithic data model encompassing the whole lifecycle. Instead, many separate models are generated. In the context of IFC, a View is a defined as a subset of the IFC Object Model that a number of implementers have agreed to support in their implementations. The software certification process is conducted according to IFC Views. Depending on agreement many IFC Views can exist with partially overlapping content or with

entirely different contents. The data exchange between applications should occur within the scope of a specific View. The entire facility lifecycle is represented across multiple Views.

The IFC standard relates to the representation of a particular instance of the facility, its components, properties, and relationships. Using the vocabulary of object-oriented modeling it can be said that it deals with object instances. It does not allow representation of the object classes and their relationships (i.e., that part is covered by the International Framework for Dictionaries (IFD), registered as the ISO 12006-3:2007 standard (IFD 2008)).

IFD is the classification system for all information in the AEC/FM field. It is an object-oriented framework that defines objects, collections and their relationships. It is intended to work as the overarching structure that will provide support for the development of the unified AEC/FM vocabularies at the national, regional or domain levels. Since all share the same structure it will be possible to translate terms between languages and domains, preferably using automated softare agents. IDF identifies each object in the model and this provides the capability to define context within which a concept is going to be used. Each object can have multiple names providing for the definition of synonyms or usage in different languages. An object is related to a formal classification system using references. The standard supports the following types of objects: Subjects, Activities, Actors, Units, Measures with Units, and Properties. Relationships are divided into: Association, Collection, Specialization, Composition, Involvement, Property Assignment, Sequence and Measure Assignment. Using these mechanisms the user can create a model-based definition of all concepts in AEC/FM including facts about classification systems, information models, object models, and process models. In other words, IFD functions as the IFC metamodel. Also IFD provides a unique global reference for any AEC/FM concept. The mechanism that relates IFC and IFD standards is scheduled to be published in the IFC 2x4 standard revision. The actual realization of IFD is the IFD Library, an international initiative currently run by four nations: Canada, Netherlands, Norway and USA. The purpose of the library is to provide semantic knowledge to the construction industry in a global and uniform way.

In addition to the above described standards a second type of interoperability formats has been developed based on another open standard - eXtensible Markup Language (XML). XML is a general-purpose specification capable of describing published data. The data description mechanism is based on the insertion of tags in the traditional text and the user can choose any term to define a particular tag. The language permits representation of arbitrary data arranged as an hierarchical tree with one element serving as the tree root (Harold and Means 2004). XML enables the structured representation of any kind of information but does not provide any mechanism to infer the meaning of the terms used in tags. One approach to the definition of a tag's meaning is the XML schema. It is a language that provides a description of a type of XML document, usually articulated in terms of constraints on the structure and content of related XML documents. Many schemas have been developed for the AEC/FM field. The gbXML (Green Building XML) schema is used for describing data relating to the building energy efficiency of the facility and its impact on the environment. The aecXML schema is used for depicting all building data in design, engineering and construction disciplines, and the CityGML schema is

used for geo-spatial data representation. Also IFC data can be represented with the ifcXML schema. Since the IFD is an EXPRESS model, the EXPRESS to XML Schema Converter (EXC 2002) can be used to obtain the XML schema for IFD.

Open standards developed for the AEC/FM industry relate to the problem of interoperability since this is the most obvious obstacle in the industry. These standards enable the highly structured representation of information about buildings but do not consider the problem of information reuse outside of the context of a particular facility lifecycle or the automatic creation of new information for later reuse. Fortunately, since all described standards have suitable XML schemas some knowledge management can be achieved with technologies developed under the Semantic Web initiative.

Semantic Web, ontology and knowledge management

The recent advent of the Semantic Web has given a new impulse to the old knowledge management research field. The goal of the endeavor is to build a unified information medium that is both understandable for people and computers, and that can be used for the automatic deduction of meaningful inferences (Berners-Lee et al. 2001). To function effectively, the Semantic Web should be built on structured collections of information and sets of inference rules. Research on knowledge representation conducted as the part of the long time effort to build artificial intelligence systems has already developed many useful technologies. Unfortunately, those systems have been centralized, relying on limited sets of rules to describe narrowly defined domains making the reuse of rules in new domains impractical. Similar to the hypertext concept, when existing knowledge representation concepts were coupled with the global information system the full potential of the technology was realized and this spurred a new wave of interest in the knowledge management field.

The new attempt to create universal knowledge representations is based on the layered structure of representation standards, where the upper layer exploits functionality of lower layers and provides greater semantic expressiveness. At the bottom level resides XML. Meaning is expressed in the next layer containing the Resource Description Framework (RDF), a data model for representing information about entities in the Web. In the Semantic Web standards an entity or thing is referred as the resource. RDF achieves its functionality by using triples, a structure consisting of subject, predicate and object. This formation states that a particular thing (subject) has a property (predicate) with a particular value (object). Subject and predicate are identified by the Universal Resource Identifier (URI) and value is either URI or literal. URIs ensure that concepts are not just bare terms devised by someone, but are connected to unique definitions on the Web (Klyne and Carroll 2004). When multiple triples point to the same resource they start to form a network of information about related things. In that way information that defines a single entity is not held in one place but is spread over the Web forming a distributed web of data. However, RDF has no mechanism for determining that two or more dissimilar terms point to the same concept.

The next level of the semantic expressiveness is achieved with an ontology. In the Semantic Web domain an ontology is identified as the formal representation that defines relationships among terms. The first level of ontological functionality is achieved with the RDF Schema (RDFS). Like other schema languages RDFS provides information about basic RDF structures. It accomplishes this task by supplying constructs that allow the declaration of classes, subclasses, property and subproperty relationships among resources. Constructs domain and range describe the relationship between properties and classes. These definitions are expressed using RDF triples (Allemang and Hendler 2008). RDFS provides a limited set of inference rules that are restricted to the transitive closure of the hierarchies.

The Web Ontology Language (OWL) currently provides the highest level of ontological functionality among Semantic Web technologies. It is a family of languages based on two semantics. OWL Lite and OWL DL are based on Description Logic semantics that guarantee completeness of reasoning. OWL Lite is a restricted version of OWL DL and is intended as a quick migration path for thesauri and other taxonomies. OWL Full provides maximum expressiveness and the syntactic freedom of RDF, but does not support complete or efficient reasoning. The language provides constructs like class, property, property restrictions, Boolean combinations, enumerations and instances. A wide range of services like reasoners and editing tools enable users to express and test knowledge using this formalism leading to the widespread acceptance of this technology (McGuinness and van Harmelen 2004).

The level of expressiveness and functionality realized in the Semantic Web development surpasses previous attempts to model computer-based knowledge management systems. It is the idea of an open community, essential to the notion of the Web, which attracts so many people to the field and generates so many results. Anyone can use open standards to develop personal systems, use open source software to express his/her knowledge, or can engage in the development of standards. This openness of the development process has resulted in the remarkable range and richness of topics covered by the Semantic Web. And this organic development that grows from the interests and energy of the supporting community persuades an increasing number of researcher to adopt both Web standards and the open development principle as a foundations for development in their professional domains.

Semantic Web based knowledge management in AEC

The easier technique to extend existing AEC standard formats and enable knowledge management functionality is to add semantic annotations using RDF. The method is demonstrated in the system for conformity checking in construction (Yurchyshyna et al. 2008). The norms are extracted from the electronic regulations and formalized as SPARQL queries in terms of the IFC model. The conformity checking process is based on matching an RDF representation of a project to a SPARQL conformity query. The project's RDF representation is extracted from the ifcXML schema and later manually enriched with domain knowledge.

More projects are using OWL to add knowledge management functionality. One notable example is the Sydney Opera House facility management model (Schevers et al. 2007). The basic IFC model is transformed using an IFC-OWL converter (Schevers and Drogemuller 2005). Existing tools are used to manually enrich this OWL representation with an ontology and rules. The OWL representation is associated with the IFC model enabling the publication of performance by selecting spaces in the 3D model.

The InteliGrid project (Dolenc et al. 2007) also uses a custom developed IFC-OWL transformer (Beetz et al. 2005) to enable the representation of expertise in addition to the basic representation of building elements and services.

The SWOP project (Böhms et al. 2008) uses a custom developed Product Modeling Ontology (PMO) that is sufficiently rich to represent a product ontology for any parametric product type. PMO is a layer on top of the RDF, RDFS and OWL hierarchy and models the product from the ontology point of view, meaning that an IFC model or any other product model can be obtained automatically as the result of the modeling process.

Today many open source converters for Semantic Web formats can be found. XSD2OWL provides transformation of an XML Schema into an OWL ontology and XML2RDF enables transformation of XML into RDF (ReDeFer 2009). These tools together with the availability of many open source editors and development environments that support Semantic Web standards provide an opportunity to add meaning and enrich open AEC standards.

Conclusion

Integration of knowledge management capabilities with BIM methodology has great potential, especially since the current open standards approach shares the same technological background as the recent Semantic Net initiative. The openness of both activities motivates many researchers and software developers to join this initiative and make their contributions. This process guaranties the best adaptation of the technologies to the users' needs and their widest support and endorsement. However, the approach also has its drawbacks. The number of technologies and their variants is massive. This overview describes just selected technologies on which we decided to base our knowledge management extension to the IFC standard after spending much time on reviewing the capabilities of many competing technologies. Moreover, the pace of change is so rapid that systems based on today's most recent technology become outdated over a two to three-year period, requiring developers to constantly update their products to take advantage of the latest version of the applied open technology.

Acknowledgments

This work was supported by the Ministry of Science and Technological Development - Republic of Serbia under grant TP - 16025. It is a part of the project 'Pilot Project: Application of the IFC (ISO/PAS 16739:2005) Standard in Drywall Systems.' The project director is Dr Igor Svetel.

References

Allemang D. and Hendler J.A. (2008); Semantic web for the working ontologist modeling in RDF, RDFS and OWL; Morgan Kaufmann, Burlington.

Beetz, J., van Leeuwen, J. P., and B. de Vries (2005); An Ontology Web Language Notation of the Industry Foundation Classes; Proceedings of the 22nd CIB W78 Conference on Information Technology in Construction; (eds. R.J., Scherer, P., Katranuschkov and S.-E. Schapke) CIB publication no. 304, Dresden (pp. 193-198)

Berners-Lee, T., Hendler, J., and O. Lassila (2001); The Semantic Web; Scientific American, Vol. 284, No. 5, (pp. 34-43)

Björk, B.-C. (1989); Basic structure of a proposed building product model; Computer-Aided Design, vol. 21, no. 2, (pp. 71-78)

Böhms, M. et al. (2008); The SWOP Semantic Product Modelling Approach; STRP NMP2-CT-2005-016972 report, Available from http://www.swop-project.eu/swop-solutions/semantic-product-modelling-approach/SWOP_D23_WP2_T2300_TNO_2008-04-15_v12.pdf (May 2009)

Dolenc, M., et al. (2007); The InteliGrid Platform for Virtual Organisations Interoperability; Itcon, Vol. 12, (pp. 459- 477)

Eastman, C.M., Bond, A. and S. Chase (1991); A Data Model for Engineering Design Databases; Artificial Intelligence in Design '91, (ed. J.S. Gero), Butterworth-Heinemann (pp. 339-365)

EXC (2002); EXPRESS to XML Schema Converter; Available from http://cic.vtt.fi/projects/ifcsvr/index_exc.html (May 2009)

Harold, E.R., Means, W.S. (2004); XML in a Nutshell; Third Edition, O'Reilly Media, Inc., Sebastopol, CA.

IFD (2008); IFD Library White Paper; Available from http://www.ifd-library.org/images/IFD_Library_White_Paper_2008-04-10_I_.pdf

Klyne, G. and J.J., Carroll (eds.) (2004); Resource Description Framework (RDF): Concepts and Abstract Syntax; W3C Recommendation 10 February 2004, Available from http://www.w3.org/TR/rdf-concepts/

Lee, G., Sacks, R., and C.M., Eastman (2006); Specifying parametric building object behavior (BOB) for a building information modeling system; Automation in Construction, Vol. 15, No. 6, (pp. 758-776)

Liebich, T., et al. (eds.) (2007); IFC2x Edition 3 TC1; International Alliance for Interoperability. Available from http://www.iai-tech.org/ifc/IFC2x3/TC1/html/index.htm

McGuinness D.L., van Harmelen F., (eds.) (2004); OWL Web Ontology Language Overview, W3C Recommendation 10 February 2004, Available from http://www.w3.org/TR/owl-features/.

ReDeFer (2009); ReDeFer - compendium of RDF-aware utilities; Available from http://rhizomik.net/redefer (May 2009)

Schevers, H., et al. (2007); Towards Digital Facility Modelling for Sydney Opera House Using IFC and Semantic Web Technology; Itcon, Vol. 12, (pp. 347- 362)

Schevers, H. and R. Drogemuller (2005); Converting IFC Data to the Web Ontology Language; Proc. International Conference on Semantics, Knowledge and Grid, Beijing, China (pp. 73-73)

Yurchyshyna, A., et al. (2008); Towards the Knowledge Capitalisation and Organisation in the Model of Conformity-Checking Process in Construction; Knowledge-Based Intelligent Information and Engineering Systems, KES 2008, Part I, vol. 5177 (eds. I. Lovrek, R.J. Howlett, and L.C. Jain), Springer, (pp. 341–348)

A Scheme and Formal Tools for Transforming the Existing Web into Semantic Web of a New Generation

V. A. Fomichov

Department of Innovations and Business
in the Sphere of Informational Technologies
Faculty of Business Informatics
State University – Higher School of Economics
Kirpichnaya str. 33, 105679 Moscow, Russia
E-mail: vdrfom@aha.ru and vfomichov@hse.ru
Fax: +7-495-771-3238

Abstract

This paper indicates the basic model of the theory of K-representations (knowledge representations) elaborated by the author as a source of ideas for constructing an advanced language platform for Semantic Web. It is to allow for modeling a system of operations on conceptual structures enabling us to build semantic representations of practically arbitrary texts in Natural Language (NL) pertaining to arbitrary field of professional activity. The next subject of the paper is a new scheme of transforming the existing Web into a Semantic Web of a new generation with the well developed mechanisms of understanding natural language texts. For the realization of this scheme, the theory of K-representations provides a number of broadly applicable formal tools. The third subject of this paper is a description of the correspondence between the input texts of two elaborated algorithms of semantic-syntactic analysis and the semantic representations of these texts being the expressions of SK-languages (standard knowledge languages). The first algorithm has a program implementation in PHP. It is a computer program transforming the definitions of a part of the notions into the OWL-expressions The forth subject of this paper is a short description of the transformations fulfilled by this program.

Keywords

Semantics-oriented natural language processing; semantic representation; theory of K-representations; formal model of a linguistic database; SK-languages; algorithm of semantic-syntactic analysis

Introduction

The immense social significance of the Web has caused the awareness of the necessity to create the Web Science. One of the central problems of Web Science is to elaborate the theory of transforming the existing Web into a Semantic Web (Berners-Lee et al. 2006; Hendler et al. 2008).

It has been possible to observe the permanent expansion in the scientific literature of the following opinion: a Semantic Web satisfying the initial goal of this project will be created in an evolutionary

way as a result of the efforts of many research groups in various fields. This point of view is expressed, in particular, in (Shadboldt et al. 2006). In this paper, the e-science international community is indicated as a community playing now one of the most important roles in quick generation of semantic content in a number of fields. The activity of this community seems to give a sign of future success of Semantic Web project.

In (Shadbolt et al. 2006), the authors ground the use of RDF as the basic language of the Semantic Web project with the help of the principle of least power: "the less expressive the language, the more reusable the data". However, it seems that the stormy progress of, first of all, e-science urges us to find a new interpretation of this principle in the context of the challenges faced nowadays by the Semantic Web project. E-science (in particular, bio informatics) needs to store on the Web the semantic content of the definitions of numerous notions, the content of scientific articles, technical reports, etc. The similar requirements are associated with semantics-oriented computer processing of the documents pertaining to economy, technology, medicine, law, politics. In particular, it is necessary to store the semantic content of the articles from newspapers, of TV-presentations, etc.

The substantial discussions of the role of semantics-oriented natural language processing mechanisms for constructing a Semantic Web satisfying the demands of numerous end users can be found in the papers (Fomichov 2000, 2008) and in the monographs (Wilks and Brewster 2006; Fomichov 2010).

That is why it can be conjectured that, in the context of the Semantic Web project, the following new interpretation of the principle of least power is reasonable: an advanced language platform for Semantic Web is to allow for modeling a system of operations on conceptual structures enabling us to build semantic representations (SRs) of practically arbitrary texts in Natural Language (NL) pertaining to arbitrary field of professional activity.

Following (Fomichov 2009), this paper indicates the basic model of the theory of K-representations (knowledge representations) elaborated by the author as a source of ideas for constructing an advanced language platform for Semantic Web. *The next subject of the paper* is a new scheme of transforming the existing Web into a Semantic Web of a new generation with the well developed mechanisms of understanding natural language texts.

For the realization of this scheme, the theory of K-representations provides a number of broadly applicable formal tools. *The third subject of this paper* is the description of the correspondence between the input texts of two elaborated algorithms of semantic-syntactic analysis and the semantic representations of these texts being the expressions of SK-languages (standard knowledge languages). The input texts of the second algorithms may belong to the sublanguages of English, German, and Russian languages. The first algorithm has a program implementation in PHP. It is a computer program transforming the definitions of the notions (these definitions form a subset of all definitions in the Russian language) into the OWL-expressions (as it is known, OWL is the principal language for developing ontologies under the framework of the Semantic Web project). *The forth subject of this paper* is a short description of the transformations fulfilled by this program.

Conceptual Operations Introduced by the Theory of K-representations

The question immediately emerges what a system of operations on conceptual structures satisfying the mentioned requirement might look like. A possible answer to this question is given by the theory of K-representations (knowledge representations) stated in numerous publications of the author in English and Russian, in particular, in (Fomichov 1996 - 2010). The basic mathematical model of this theory describes a system consisting of 10 partial operations on conceptual structures (Fomichov 2002b, 2002c, 2005a, 2007, 2010). The model determines a new class of formal languages for building SRs of sentences and complex discourses in NL – the class of SK-languages (standard knowledge languages). An early version of this model set forth in (Fomichov 1996 - 1998) determines the class of RSK-languages (restricted standard knowledge languages).

Let's consider the central ideas of determining the class of SK-languages At the first step (consisting of a rather long sequence of auxiliary steps), a class of formal objects called *conceptual bases (c.b)* is defined. Each c.b. *B* is equivalent to a system of the form $(c_1, ..., c_{15})$ with the components $c_1,..., c_{15}$ being mainly finite or countable sets of symbols and distinguished elements of such sets. In particular, $c_1 = St$ is a finite set of symbols called sorts and designating the most general considered notions (concepts); $c_2 = P$ is a distinguished sort "meaning of proposition"; $c_4 = X$ is a countable set of strings used as elementary blocks for building knowledge modules and semantic representations (SRs) of texts; *X* is called a primary informational universe; $c_5 = V$ is a countable set of variables; $c_7 = F$ is a subset of *X* whose elements are called functional symbols.
Each c.b. *B* determines three classes of formulas, the first class *Ls(B)* being considered as the principal one and being called *the SK-language (standard knowledge language) in the basis B*. Its strings (they are called K-strings) are convenient for building SRs of NL-texts. We'll consider below only the formulas from the first class *Ls(B).*

In order to determine for arbitrary c.b. *B* three classes of formulas, a collection of inference rules P[0], P[1],..., P[10] is defined. The rule P[0] provides an initial stock of formulas from the first class. E.g., there is such c.b. B_1 that, according to P[0], $Ls(B_1)$ includes the elements

house1, blue, city, set, France, 17, all, any,
Weight, Distance, Staff, Suppliers, Quantity, x1, x7, P1, P3.

For arbitrary c.b. *B*, let *Degr(B)* be the union of all Cartesian m-degrees of *Ls(B),* where $m \geq 1$.
Then the meaning of the rules of constructing well-formed formulas P[0], P[1],..., P[10] can be explained as follows: for each k from 1 to 10, the rule P[k] determines a partial unary operation *Op[k]* on the set *Degr(B)* with the value being an element of *Ls(B).*

For instance, there is such conceptual basis B that the value of the partial operation *Op[7]* (it governs the use of logical connectives AND and OR) on the four-tuple
< *v , Austria, France, Germany* > is the K-string *(Austria v France v Germany).*

Thus, the essence of the basic model of the theory of SK-languages is as follows: this model determines a partial algebra of the form

(*Degr(B), Operations(B)*) ,

where *Degr(B)* is the carrier of the partial algebra, *Operations(B)* is the set consisting of the partial unary operations *Op[1], ..., Op[10]* on *Degr(B)*.

The volume of complete descriptions in (Fomichov 2005a, 2007, 2010) of the mathematical model introducing, in essence, the operations *Op[1], ..., Op[10]* on *Degr(B)* and, as a consequence, determining the class of SK-languages considerably exceeds the volume of this paper. That is why, due to objective reasons, this model can't be included in this paper. The short characteristics of thes partial operations on conceptual structures can be found, in particular, in (Fomichov 2002a, 2005b, 2008).

The Context for the Proposed Scheme of Developing Semantic Web of a New Generation

During several last years, it has been possible to observe that the achieved state of Semantic Web and a state to be relatively soon achieved are considerably different from the state of affairs outlined as the goal in the starting publication on Semantic Web by T. Berners-Lee, J. Hendler, and O. Lassila (Berners-Lee et al. 2001).

The principal reason for this conclusion is the lack of large-scale applications implemented under the framework of Semantic Web project. This situation is implied by the lack of a sufficiently big amount (of "a critical mass") of formally represented content conveyed by numerous informational sources in many fields. This means the lack of a sufficiently big amount of Web-sources and Web-services with semantic annotations, of the visual images stored in multimedia databases and linked with the high-level conceptual descriptions, rich ontologies, etc.

This situation is characterized in the Call for Papers of the First International Symposium on Incentives for Semantic Web (Germany, Karlsruhe, October 2008) as *the lack of a critical mass of semantic content.*

That is why it has been possible to observe the permanent expansion in the scientific literature of the following opinion: a Semantic Web satisfying the initial goal of this project will be created in an evolutionary way as a result of the efforts of many research groups in various fields. In particular, this opinion is expressed in (Angelova 2003).

It is important to underline that this point of view is also expressed in the article "Semantic Web Revisited" written by the pioneers of Web: N. Shadboldt, W. Hall, T. Berners-Lee (Shadboldt et al. 2006). In this paper, the e-science international community is indicated as a community playing now one of the most important roles in quick generation of semantic content in a number of fields. The activity of this community seems to give a sign of future success of Semantic Web project.

One of the brightest manifestations of the need for new, strong impulses to developing Semantic Web is the organization of the First International Symposium on Incentives for Semantic Web under the framework of the Semantic Web International Conference - 2008.

The content of the next section is to be considered in the context of the broadly recognized need for the incentives for Semantic Web, in particular, for the incentives on the models stimulating the development of Semantic Web.

A Possible Scheme of Developing Semantic Web of a New Generation

The analysis of the expressive power of the class of SK-languages allowed for putting forward the following hypothesis in (Fomichov 2002c, 2005a, 2007): SK-languages are a convenient tool of building semantic representations of arbitrary complex natural language texts (sentences and discourses) pertaining to arbitrary field of professional activity.

Let's consider the principal ideas of a new, theoretically possible scheme aimed at transforming the existing Web into a Semantic Web. The proposed scheme is based on (a) the mathematical model constructed in (Fomichov 2005a, 2007, 2010) and describing a system of 10 partial operations on conceptual structures and (b) the analysis of the expressive mechanisms of SK-languages carried out in the mentioned works. The new scheme can be very shortly formulated as follows:

1. An XML-based format for representing the expressions of SK-languages (standard knowledge languages) will be elaborated. Let's agree that the term "a K-representation of a NL-text T" means below a semantic representation of T built in this format and that the term "a semantic K-annotation" will be interpreted below as a K-representation of a NL-annotation of an informational source. The similar interpretations will have the terms "a K-representation of a knowledge piece" and "a high-level conceptual K-description of a visual image".
2. The NL-interfaces for different sublanguages of NL (English, Russian, German, Chinese, Japan, etc.) helping the end users to build semantic K-annotations of Web-sources and Web-services are being designed.
3. The advanced ontologies being compatible with OWL and using K-representations of knowledge pieces are being elaborated.
4. The new content languages using K-representations of the content of messages sent by computer intelligent agents (CIAs) in multi-agent systems are being worked up. In particular, this class of languages is to include a subclass being convenient for building the contracts concluded by the CIAs as a result of successful commercial negotiations.
5. The visual images of the data stored in multimedia databases are being linked with high-level conceptual K-descriptions of these images.
6. The NL-interfaces transforming the NL-requests of the end users of Web into the K-representations of are being designed.
7. The advanced Web-based search and question-answering systems are being created being able (a) to transform (depending on the input request) the fragments of a discourse into the K-representations, (b) to analyze these K-representations of the discourse fragments, and (c) to analyze semantic K- annotations of Web-sources and Web-services.
8. The NL processing systems being able to automatically extract knowledge from NL-texts, to build the K-representations of knowledge pieces, and to inscribe these K-representations into the existing ontologies are being elaborated.
9. The generators of NL-texts (the recommendations for the users of expert systems or of recommender systems, the summaries of Web-documents, etc.) using the SK-languages for

representing the meaning of a NL-text to be synthesized are being constructed. Besides, a reasonable direction of research seems to be the design of applied intelligent systems being able to present the semantic content of a message for the end user as an expression of a non-standard K-language being similar to a NL-expression but containing, may be, a number of brackets, variables, markers.

Fulfilling these steps, the international scientific community will create in a reasonable time a digital conceptual space unified by a general-purpose language platform. The realization of this scheme will depend on the results of its discussion by the international scientific community.

The Formal Tools Provided by the Theory of K-representations

The monographs (Fomichov 2005a, 2010) stating two versions of the theory of K-representations propose one universal (most likely) and several broadly applicable formal tools for the realization of this scheme.

The theory of K-representations is an expansion of the theory of K-calculuses and K-languages (the KCL-theory). The basic ideas and results of the KCL-theory are reflected in numerous publications both in Russian and English, in particular, in (Fomichov 1996 – 2005a).

The *first basic constituent* of the theory of K-representations is the theory of SK-languages (standard knowledge languages), stated, in particular, in (Fomichov 1996 – 2005b). The kernel of the theory of SK-languages is a mathematical model describing a system of such 10 partial operations on structured meanings (SMs) of natural language texts (NL-texts) that, using primitive conceptual items as "blocks", we are able to build SMs of arbitrary NL-texts (including articles, textbooks, etc.) and arbitrary pieces of knowledge about the world. The outlines of this model can be found in two papers published by Springer in the series “Lecture Notes in Computer Science” (Fomichov 2002a, 2005b).

The analysis of the scientific literature on artificial intelligence theory, mathematical and computational linguistics shows that today the class of SK-languages opens the broadest prospects for building semantic representations (SRs) of NL-texts (i.e., for representing meanings of NL-texts in a formal way).

The expressions of SK-languages will be called below the K-strings. If T is an expression in natural language (NL) and a K-string *E* can be interpreted as a SR of T, then *E* will be called a K-representation (KR) of the expression T.

The *second basic constituent* of the theory of K-representations is a widely applicable mathematical model of a linguistic database (LDB). The model describes the frames expressing the necessary conditions of the existence of semantic relations, in particular, in the word combinations of the following kinds: “Verbal form (verb, participle, gerund) + Preposition + Noun”, “Verbal form+ Noun”, “Noun1 + Preposition + Noun2”, “Noun1+ Noun2”, “Number designation + Noun”, “Attribute+Noun”, “Interrogative word + Verb”. The expressive power of SK-languages enables us to associate the lexical units with the appropriate simple or compound

semantic units. The model describes the logical structure of linguistic databases being the components of natural-language interfaces to intelligent databases as well as to other applied computer systems.

The *third basic constituent* of the theory of K-representations is two complicated, strongly structured algorithms carrying out semantic-syntactic analysis of texts from some practically interesting sublanguages of NL. These algorithms, called *SemSyn* and *SemSynt1* respectively, are based on the elaborated formal model of a linguistic database. The algorithm *SemSyn* transforms a NL-text in its semantic representation being a K-representation, the algorithm *SemSyn* is described in two final chapters of the monograph (Fomichov 2005a), and the algorithm *SemSynt1* is set forth in Chapters 9 and 10 of the monograph (Fomichov 2010).

An important feature of these algorithms is that they don't construct any syntactic representation of the inputted NL-text but directly finds semantic relations between text units. Since numerous lexical units have several meanings, the algorithm uses the information from a linguistic database and linguistic *context* for choosing one meaning of a lexical unit among several possible meanings.

The other distinguished feature is that these complicated algorithms are completely described with the help of formal tools, that is why they are problem independent and don't depend on a programming system. The algorithm *SemSyn* is implemented in the Web programming language PHP.

The Principles of Selecting the Form of a Text's Semantic Representation

Let's consider the examples illustrating the correspondence between the natural language sentences in English and their semantic representations (SR) being the expressions of a certain SK-language, that is, being the K-representations of the input texts. In these examples, the SR of the input text T will be the value of the string variable *Semrepr (Semantic representation).*

Example 1. Let T1 = "The writer Igor Somov lives in Saratov". Then
Semrepr = Situation (e1, living * (Time, #now#)
(Agent1, certn person * (Qualification, writer)
(Name, 'Igor') (Surname, 'Somov') : x2)
(Place1, certn city * (Name1, 'Saratov') : x3)).

Example 2. Let T2 = "Deliver a box with details to the warehouse _ 3". Then
Semrepr = (Command (#Operator#, #Executor#, #now#, e1)
∧ Target (e1, delivery1 * (Object1, certn box1 *
(Content1, certn set * (Qual-compos, detail)) : x1)
(Place2, certn warehouse * (Number1, 3) : x2).)

Example 3. Let T3 = "Did the international scientific conference "COLING" take place in Asia?". Then
Semrepr = Question (x1, (x1 _
Truth-value (Situation (e1, taking_place *

(Time, certn moment * (Earlier, #now#) : t1)
(Event1, certn conference * (Type1, international)
(Type2, scientific) (Name1, 'COLING') : x2)
(Place, certn continent * (Name, 'Asia') : x3))))).

Example 4. Let T4 = "What publishing house has released the novel "Winds of Africa"?". Then
Semrepr = Question (x1, Situation (e1, releasing1 *
(Time, certn moment * (Earlier, #now#) : t1)
(Agent2, certn publ-house : x1)
(Product1, certn novel1 * (Name1, 'Winds of Africa') : x2))).

Example 5. Let T5 = "What foreign publishing houses the writer Igor Somov is collaborating with?". Then
Semrepr = Question (S1, (Qual-compos (S1, publish-house *
(Type-geographic, foreign)) ∧
Description(arbitrary publish-house * (Element, S1) : y1,
Situation (e1, collaboration * (Time, #now#)
(Agent1, certn person * (Occupation, writer)
(Name, 'Igor') (Surname, 'Somov') : x1)
(Organization1, y1))))).

Example 6. Let T6 = "Who produces the medicine "Zinnat"?". Then
Semrepr = Question (x1, Situation (e1, production1 *
(Time, #now#) (Agent2, x1)
(Product2, certn medicine1 * (Name1, 'Zinnat') : x2))).

Example 7. Let T7 = "For whom and where the three-ton aluminum container has been delivered from?".
Semrepr = Question ((x1 ∧ x2),
Situation (e1, delivery2 *
(Time, certn moment * (Earlier, #now#) : t1)
(Recipient, x1) (Place1, x2)
(Object1, certn container1 * (Weight, 3/ton)
(Material, aluminum) : x3))).

Example 8. Let T8 = "How many people did participate in the creation of the textbook on statistics?". Then
Semrepr = Question (x1, ((x1 _ Numb(S1))
∧ Qual-compos (S1, person) ∧
Description (certn person * (Element, S1) : y1,
Situation (_1, participation1 *
(Time, certn moment * (Earlier, #now#) : t1)
(Agent1, y1) (Type-of-activity, creation1 *
(Product1, certn textbook * (Area1, statistics) : x2))))).

Example 9. Let T9 = "How many times Mr. Stepan Semenov flew to Mexico?". Then

Semrepr = Question (x1, ((x1 _ Numb (S1))
∧ Qual-compos (S1, sit) ∧
Description (arbitrary sit * (Element, S1) : e1,
Situation (e1, flight * (Time, certn moment *
(Earlier, #now#) : t1) (Agent1, certn person *
(Name, 'Stepan')(Surname, 'Semenov') : x2)
(Place2, certn country * (Name, 'Mexico') : x3))))).

A Transformer of Natural Language Knowledge Descriptions into OWL-expressions

In the context of transforming step by step the existing Web into Semantic Web (see Fomichov 2008, 2009), the need for large Web-based and interrelated collections of formally represented pieces of knowledge covering many fields of professional activity is a weighty ground for increasing the interest of the researchers to the problem of automated formation of ontologies.

It seems that the most obvious and broadly applicable way is to construct a family of natural language processing systems being able to transform the descriptions of knowledge pieces in NL (in English, Russian, German, French, Chinese, Japanese, etc.) into the OWL-expressions and later, possibly, into the expressions of an advanced formalism for developing ontologies.

This idea underlay the design of the computer system NL-OWL1, it is a Russian-language interface implementing a modification of the algorithm of semantic-syntactic analysis *SemSyn* stated in (Fomichov 2005a). The main directions of expanding the input language of the algorithm *SemSyn* are as follows:
- the definitions of notions in restricted Russian language can be the input texts of the system;
- a mechanism of processing the homogenious members of the sentence is added to the algorithm of semantic-syntactic analysis;
- a part of input sentences (the descriptions of the events and the definitions of notions) is transformed not only into the K-representations (i.e. into the expressions of a certain SK-language) but also, at the second stage, into the OWL-expressions.

Let's consider the examples illustrating the principles of processing NL-texts by the experimental Russian-language interface NL-OWL1, implemented in the Web programming system PHP.

Example 1. Definition: "Carburettor is a device for preparing a gas mixture of petrol and air"

K-representation:

ModuleOfKnowledge (definition; carburettor; x1;
(Definition1 (certn carburettor : x1, certn device : x2) ∧
Purpose (x2, certn preparation1 : x_e1) ∧
Description (preparation1, Object1 (certn mixture *
(Type, gas) : x3)) ∧ Product1 (x3, certn petrol : x4)
∧ Product1 (x3, air))))

OWL-expression:

< owl Class rdf ID = "ModuleOfKnowledge"/>

< hasFormModule rdf resource = "#definition"/>

< hasConcept rdf resource = "#carburettor_x1"/>

< owl Class rdf ID = "Action"/>

< Action rdf ID = "Concept"/>

< owl Class rdf ID = "Situation"/>

<Situation rdf ID = "x1">

< has Action rdf resource = "#Concept"/>

< hasDetermination rdf resource = "#Device_x2"/>

</Situation>

< owl Class rdf ID = "Device"/>

< Device rdf ID = "Device_x2"/>

< Destination rdf resource = "#preparation_x_e1"/>

</Device>

< owl Class rdf ID = "Destination"/>

< Destination rdf ID = "Destination_x_e1"/ >

< Object1 rdf resource = "#Mixture_x3"/>

</Destination>

< owl Class rdf ID = "Mixture"/>

< Mixture rdf ID = "Mixture_x3"/>

< Form rdf resource = "#gas"/>

< Product1 rdf resource = "#Petrol_x4"/>

< Product1 rdf resource = "#Air_x5"/>

</Mixture>

Due to the broad expressive possibilities of SK-languages, the intelligent power of the transformer NL-OWL1 can be considerably enhanced. That is why the formal methods underlain the design of the system NL-OWL1 enrich the theoretical foundations of the Semantic Web project.

References

1. Angelova, G. (2005); Language technology meet ontology acquisition. In: Dau, F., Mugnier, M.-L., Stumme, G. (eds.) International Conference on Conceptual Structures 2005. LNCS 3596, Springer, Heidelberg (pp. 367-380).
2. Berners-Lee, T., Hendler, J. and O. Lassila (2001); The Semantic Web; Scientific American (pp. 34-43)
3. Berners-Lee, T., Hall, W., Hendler, J.A., O'Hara, K., Shadbolt, N. and D.J. Weitzner (2006); A Framework for Web Science; Foundations and Trends in Web Science, Vol. 1, No. 1, now Publishers Inc. (pp. 134)
4. Fomichov, V.A. (1996); A mathematical model for describing structured items of conceptual level; Informatica. An Intern. J. of Computing and Informatics (Slovenia), 1996, Vol. 20, No, 1. P. 5-32.
5. Fomichov, V.A. (1998); Theory of Restricted K-calculuses as a Comprehensive Framework for Constructing Agent Communication Languages; Special Issue on NLP and Multi-Agent Systems. In: Fomichov V.A., Zeleznikar A.P. (eds.). Informatica. An International Journal of Computing and Informatics (Slovenia). Vol. 22, No 4 (pp. 451-63)
6. Fomichov, V.A. (2000); An ontological mathematical framework for electronic commerce and semantically-structured Web. In: Zhang Y., Fomichov V.A., Zeleznikar A.P. (eds.), Special Issue on Database, Web, and Cooperative Systems. Informatica. An International Journal of Computing and Informatics (Slovenia, Europe), Vol. 24, No. 1, 39--49 (2000)
7. Fomichov, V.A. (2002a); Theory of K-calculuses as a Powerful and Flexible Mathematical Framework for Building Ontologies and Designing Natural Language Processing Systems (ed. Troels Andreasen, Amihai Motro, Henning Christiansen, Henrik Legind Larsen), Flexible Query Answering Systems. 5th International Conference, FQAS 2002, Copenhagen, Denmark, October 27 - 29, 2002. Proceedings; LNAI 2522 (Lecture Notes in Artificial Intelligence, Vol. 2522), Springer Verlag (pp. 183-196)
8. Fomichov, V.A. (2002b); Mathematical foundations of representing texts' meanings for the development of linguistic informational technologies. Part 1. A model of the system of primary units of conceptual level. Informational Technologies (Moscow) No. 2002/10 (pp. 16-25)
9. Fomichov, V.A. (2002c); Mathematical foundations of representing texts' meanings for the development of linguistic informational technologies. Part 2. A system of the rules for constructing semantic representations of phrases and complex discourses. Informational Technologies (Moscow), No. 2002/11 (pp. 34-45)
10. Fomichov, V.A. (2005a); The Formalization of Designing Natural Language Processing Systems. Moscow, MAX Press; in Russian (pp. 368)

11. Fomichov, V.A. (2005b); Standard K-Languages as a Powerful and Flexible Tool for Building Contracts and Representing Contents of Arbitrary E-Negotiations. In: Bauknecht K., Proell B., Werthner H. (eds.), The 6th Intern. Conf. on Electronic Commerce and Web Technologies "EC-Web 2005", Copenhagen, Denmark, Aug. 23 - 26, Proceedings. Lecture Notes in Computer Science, Vol. 3590, Springer Verlag (pp. 138-47)
12. Fomichov, V.A. (2007); Mathematical Foundations of Representing the Content of Messages Sent by Computer Intelligent Agents; Moscow, State University – Higher School of Economics, The Publishing House "TEIS", 2007 (in Russian).
13. Fomichov, V.A. (2008); A comprehensive mathematical framework for bridging a gap between two approaches to creating a meaning-understanding Web; International Journal of Intelligent Computing and Cybernetics (Emerald Group Publishing Limited, UK). 2008, Vol. 1, No. 1 (pp. 143-63)
14. Fomichov, V.A. (2009); Theory of K-representations as a Source of an Advanced Language Platform for Semantic Web of a New Generation; On-line Proceedings of the First International Conference on Web Science, Athens, Greece, March 18-20, 2009; available at http://journal.webscience.org/221/1/websci09_submission_128.pdf
15. Fomichov, V.A. (2010); Semantics-Oriented Natural Language Processing: Mathemati-cal Models and Algorithms; New York, Berlin, Heidelberg, Springer US (pp. 348; in press)
16. Hendler, J., Shadboldt, N., Hall, W., Berners-Lee, T. and D. Weitzner, D. (2008); Web science: an interdisciplinary approach to understanding the web. Communications of the ACM, 2008, Vol. 51, No. 7 (pp. 60-69)
17. Shadboldt, N., Hall, W., Berners-Lee, T. (2006); Semantic Web Revisited. IEEE Intelligent Systems, Vol. 21, No. 3.
18. Wilks, Y. and C. Brewster (2006); Natural Language Processing as a Foundation of the Semantic Web; Foundations and Trends in Web Science, Vol. 1, No. 3, now Publishers Inc.

The Representation of Context in Computer Software

Hisham Assal[*], Kym Pohl[**], and Jens Pohl[*]

[*]Collaborative Agent Design Research Center, California Polytechnic State University
[**]CDM Technologies, Inc.
San Luis Obispo, California, USA

Abstract

Computers do not have the equivalent of a human cognitive system and therefore store data simply as the numbers and words that are entered into the computer. For a computer to interpret data it requires an information structure that provides at least some level of *context*. This can be accomplished utilizing an *ontology* of objects with characteristics, semantic behavior, and a rich set of relationships to create a virtual version of real world situations and provide the *context* within which *intelligent* logic (e.g., *agents*) can automatically operate.

This paper discusses the process of developing ontologies that serve to provide context for agents to interpret and reason about data changes in decision-support software tools, services and systems.

Keywords: agents, CMAP Tools, context, semantic model, data, information, information-centric, knowledge, ontology, OWL, Protégé, use-case, UML, WordNet.

The Need for Context

The design of any information system architecture must be based on the obvious truth that the only meaningful reason for capturing and storing data is to utilize them in some planning or decision-making process. However for data to be useful for planners and decision makers they have to be understood in context. In other words, data are just numbers and words that become meaningful only when they are viewed within a situational framework. This framework is typically defined by an expressive *blueprint* rich in associations that relate data items to each other and peripheral factors that influence the meaning of the data in a particular situation. Succinctly stated, numbers and words (i.e., data) found within a rich, structured set of relationships become information, which provides the necessary context for interpreting the meaning of the data, the recognition of patterns, and the formulation of rules, commonly referred to as knowledge (Pohl 2003, 1-3)

The larger an organization the more data it generates, both by itself as well as captured from external sources. With the availability of powerful computer hardware and database management systems the ability of organizations to store and order these data in some purposeful manner has dramatically increased. However, at the same time, the expectations and need to utilize the stored data in monitoring, planning and time-critical decision-making tasks has become a major human resource intensive preoccupation. In many respects this data-centric focus has become a bottleneck that inhibits the ability of the organization to efficiently and effectively accomplish its mission.

The reasons for this bottleneck are twofold. First, large organizations are forced to focus their attention and efforts on the almost overwhelming tasks involved in converting unordered data into purposefully ordered data (Figure 1). This involves, in particular, the establishment of gateways to a large number of heterogeneous data sources, the validation and integration of these sources, the standardization of nomenclatures, and the collection of data elements into logical data models.

Second, with the almost exclusive emphasis on the slicing and dicing of data, rather than the capture and preservation of relationships, the interpretation of the massive and continuously increasing volume of data is left to the users of the data (Figure 2). The experience and knowledge stored in the human cognitive system serves as the necessary context for the interpretation and utilization of the ordered data in monitoring, planning and decision-making processes. However, the burden imposed on the human user of having to interpret large amounts of data at the lowest levels of context has resulted in a wasteful and often ineffective application of valuable and scarce human resources. In particular, it often leads to late or non-recognition of patterns, overlooked consequences, missed opportunities, incomplete and inaccurate assessments, inability to respond in a timely manner, marginal decisions, and unnecessary human burn-out.

These are symptoms of an incomplete information management environment. An environment that relies entirely on the capture of data and the ability of its human users to add the relationships to convert the data into information and thereby provide the context that is required for all effective planning and decision-making endeavors.

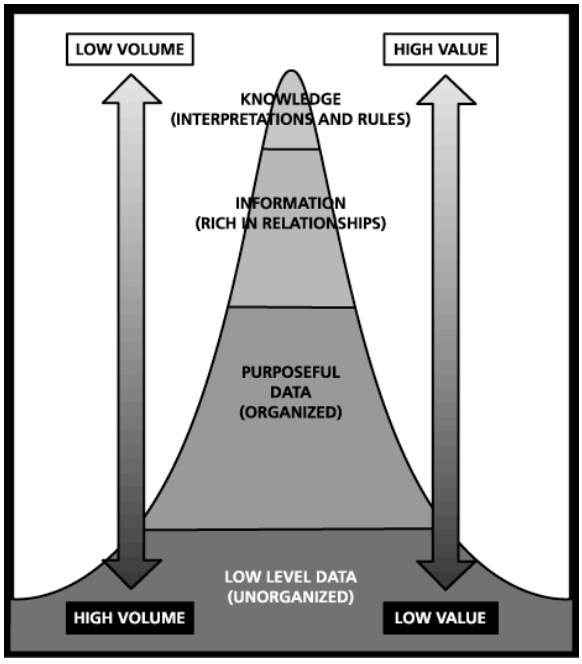


Figure 1: Transition from data to knowledge

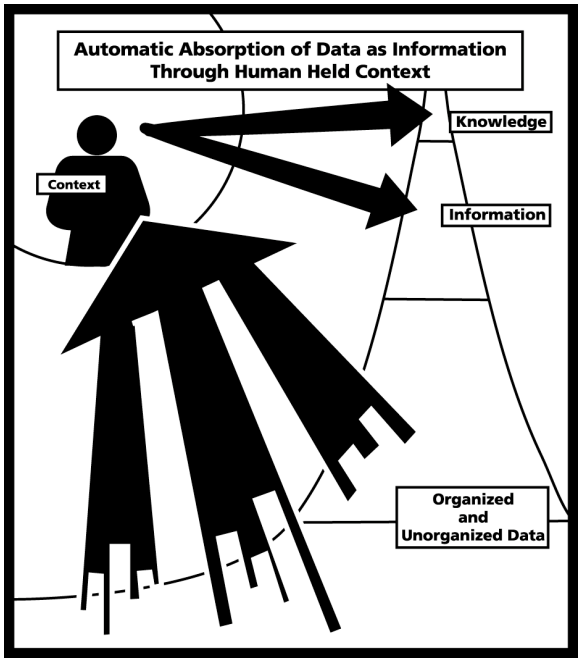


Figure 2: Human interpretation of data

From a conceptual point of view, a more complete information management environment considers data to be the bottom layer of a three-layer architecture, namely: a data layer; a mediation layer; and, an information (i.e., semantic) layer. The *Data Layer* is responsible for

integrating heterogeneous data sources into accessible and purposefully ordered data. It typically includes a wide variety of repositories ranging from simple textual files to databases, Data Portals, Data Warehouses, and Data Marts.

The *Mediation Layer* defines the structure of the data sources (i.e., logical data models), data transfer formats, and data transformation rules. The two principal purposes of the Mediation Layer are to facilitate the automated discovery of data, the reconciliation (i.e., merging) of data, and finally the mapping of such unified data to information. In other words, the Mediation Layer serves as a registry for all definitions, schemas, protocols, conventions, and rules that are required to recognize data within the appropriate *context*. The Mediation Layer also serves as a translation facility for bridging between data with structural relationships (e.g., based on a logical data model) and information that is rich in contextual relationships.

The *Information Layer* consists of many functionally oriented planning and decision-assistance software applications and services. Typically, these semantic capabilities are based on internal information models (i.e., object models or ontologies) that are virtual representations of particular portions of the real world context. By providing context, the internal information model of each application is able to support the automated reasoning capabilities of rule-based software agents.

In such a three-layered information management environment the Mediation Layer continuously populates the information models of the applications in the Information Layer with the data changes that are fed to it by the Data Layer. This in turn automatically triggers the reasoning capabilities of the software agents. The collaboration of these agents with each other and the human users contributes a powerful, near real-time, adaptive decision-support environment. The agents can be looked upon as intelligent, dynamic tools that continuously monitor changes in the real world. They utilize their reasoning and computational capabilities to generate and evaluate courses of action in response to both real world events and user interactions. As a result the human user is empowered by agent-based assistance while at the same time relieved of many of the lower level filtering, analysis, and reasoning tasks that are a necessary part of any useful planning and problem solving process. A vital enabler of these benefits is the ability of the software agents to continuously and tirelessly monitor the real world execution environment for changes and events that may impact current or projected plans.

Definition of Terms

The following brief explanation of key terms and concepts referred to in this paper is provided as an introduction for clarification purposes.

> ***Ontology:*** The term ontology is loosely used to describe an information structure, rich in relationships that provides a virtual representation of some real world environment (e.g., the context of a problem situation such as the management of a transport corridor, the loading of a cargo ship, the coordination of a military theater, the design of a building, and so on). The elements of an ontology include objects and their characteristics, different kinds of relationships among objects, and the concept of inheritance. Ontologies are also commonly referred to as *object models*. However, strictly speaking the term ontology has a much broader definition. It actually refers to the entire knowledge in a particular field. In this sense an ontology would include both an object model and the

software agents that are capable of reasoning about information within the *context* provided by the object model (since the agents utilize business rules that constitute some of the knowledge within a particular domain).

Information and context: Information refers to the combination of data with relationships to provide adequate *context* for the interpretation of the data. The richer the relationships the greater the *context* (i.e., meaning conveyed by the combination of data with relationships), and the more opportunity for automatic reasoning by software agents.

Information-centric: Software that incorporates an internal information model, such as an ontology, is often referred to as *information-centric* software. The information model is a virtual representation of the real world domain under consideration and is designed to provide adequate *context* for software agents (typically rule-based) to reason about the current state of the virtual environment. Since information-centric software has some understanding of what it is processing it normally contains tools rather than predefined solutions to predetermined problems. These tools are commonly software agents that collaborate with each other and the human user(s) to develop solutions to problems in near real-time, as they occur. Communication between information-centric applications is greatly facilitated since only the changes in information need to be transmitted. This is made possible by the fact that the object, its characteristics and its relationships are already known by the receiving application.

Agents: This term has been applied very loosely in recent years. There are several different kinds of agents. Symbolic reasoning agents are most commonly associated with knowledge management systems. These agents may be described as software modules that are capable of reasoning about events (i.e., changes in data received from external sources or as the result of internal activities) within the context of the information contained in an internal information model (i.e., ontology). The agents collaborate with each other and the human users as they monitor, interpret, analyze, evaluate, and inform users of emerging issues or plan alternative courses of action.

Identifying the Purpose of the Ontology

The objective(s), or intended purpose, of the ontology must be defined in some formal manner in order to facilitate the development process. Without a well-defined purpose of the ontology, development can continue with no apparent end-state, and the ontology can grow in different directions beyond the control of system developers. Some common purposes of ontologies include the representation of knowledge in a given domain of interest, facilitating communication among system components, re-use by other applications, or as a common language for multiple systems within the same domain.

A good way for defining the purpose of the ontology is by means of use-cases[1]. The utility an ontology must provide can be broken down into specific, well-defined use-cases, in which actors and actions are identified, as well as the perceived components that will be involved in each

[1] In software development a use-case describes a process or scenario from the point of view of the actors involved in the process (Cockburn A. (2001); 'Writing Effective Use Cases'; Addison-Wesley, Reading, Massachusetts.)

action. Another tool for identifying the purpose of an ontology is a set of questions, which the ontology should be equipped to answer.

The ontology is complete, in the context of a given set of requirements, when all the use-cases are supported by ontology concepts and all the questions needed to be asked can be answered by the ontology.

Building the Ontology

Once the domain knowledge has been captured in free form, it is time to start building the ontology. This is the step, where the captured knowledge is formalized and concepts are given specific descriptive names to allow the communication with other stakeholders. The process of building the ontology can be described in the following steps:

1. Capture the knowledge in the domain of interest. Many knowledge acquisition techniques can be applied in this step, including textbooks, interviewing subject matter experts (SME), databases of case studies, analysis reports, and so on. One of the primary methods of capturing knowledge in this domain is utilizing a subject matter expert to formalize the concepts and produce the model. Other sources of knowledge assist the expert in this task, such as books, military manuals, past plan analyses, and training material.
2. Identify the key concepts and relationships in the domain of interest. The key concepts are the ones that relate to the identified purpose of the ontology. They typically answer critical questions or contribute to the communication among system components and concepts that are involved in actions of use-cases. Other concepts that help to relate key concepts to each other or add details to key concepts, are considered supporting concepts. For example, the key concepts in a human factors ontology are likely to be: Person, Organization, Communication, Personal Traits, and Behavioral Traits.
3. Produce precise textual definitions of such concepts and relationships. The textual definitions help disambiguate the concepts and define their role in the ontology. Existing textual definitions in standard lexicons can help in this step. For example, the WordNet[2] database offers a good electronic resource for common definitions of English language terms. The use of a lexicon like WordNet also facilitates the search for terms and their synonyms, for the purpose of analyzing free text. Other specialized lexicons, such as military manuals, can also be a good source for accepted definitions for common terms.
4. Identify terms to refer to such concepts and relationships. The selection of meaningful ontology terms helps developers understand the role of each concept and possibly the common uses of it. Also system developers do not need to go back to the formal definition of each term every time they need to

[2] WordNet is an electronic lexical database developed by researchers at Princeton University (Fellbaum C. (ed.); 'WordNet: An Electronic Lexical Database', Bradford Book Series, MIT Press, Cambridge, Massachusetts.) and http://wordnet.princeton.edu/perl/webwn3.0?s=word-you-want

use it. The selected terms should be expressive of the concept and close to its natural language description.

5. Obtain agreement on all of the above. It is important for all stakeholders to agree on the selection of concepts and the terms used to refer to them in the ontology. Ontology-based systems are typically a collaborative effort, often among multiple organizations. To facilitate communication among all participants there has to be agreement on the ontology.
6. Select a representational methodology (e.g., Protégé[3], UML[4], etc.). Modeling of ontologies is a step to formalizing the captured knowledge and producing an artifact, which can communicate that knowledge to other stakeholders. Most modeling methods have a graphical notation to easily connect concepts and navigate through the ontology. The criteria for selection of a modeling method are:
    - *Coverage:* Does the crafted model provide enough elements to represent all of the captured concepts and the types of relationships that exist among them?
    - *Granularity:* How much detail can the modeler represent in a concept?
    - *Learning curve:* Is this modeling method a standard method, which modelers are already familiar with? Or is it a new method that requires investment of time and effort to learn to use efficiently?

    Protégé is the modeler of choice for OWL[5]-based ontologies. There are other tools that support OWL development, such as Concept Maps, but the support that is offered by Protégé is stronger in visualization and ontology navigation.

Coding the Ontology

The implementation of ontology-based systems requires translating the ontology model into an implementation language. The language chosen for coding an ontology (e.g., formal logic, UML, OWL, etc.) has to provide the following characteristics:

- *Conceptual distance:* The ability of the language to represent abstract concepts at multiple levels of abstraction.

---

[3] Protégé is an *Open Source* ontology editor and knowledge-base framework that is extensible and based on Java (see: http://protege.stanford.edu/users.html)

[4] UML (Unified Modeling Language) provides a standard notation for modeling systems and context utilizing object-oriented concepts and principles (Booch G., J. Rumbaugh and I. Jacobson (1999); 'The Unified Modeling Language User Guide'; Addison-Wesley, New York, New York.)

[5] OWL (Web Ontology Language) facilitates the machine processing of the content of data through software (McGuinness D. and van Harmelen F. (2004); 'OWL Web Ontology Language Overview'; MIT Press, Cambridge, Massachusetts.)

- *Expressive power:* The ability to represent complex concepts with consistent language constructs.

- *Standards compliant:* The language should follow accepted standards and notations to allow for better communication among development team members.
- *Translatability:* The language constructs have formal structures that can be converted to forms in other languages, without ambiguity.
- *Guidelines:* The model development process is supported by a set of guidelines and best practices.
- *Formal semantics:* The intended meaning of each language construct is unambiguous and well-defined.
- *Flexibility:* The ability to represent concepts in different ways, using different constructs.
- *User base:* The availability of user groups provides support for ontology development, through the exchange of experiences and best practices.
- *Availability:* The language has to be available, preferably, in the public domain, along with tools to support its use.

For example, the selected coding language may be OWL to facilitate communication with other system developers, especially in the case of a multi-organization effort. OWL satisfies many of the selection criteria mentioned above.

- *Conceptual distance:* OWL allows the representation of abstract concepts, maintaining its level of abstraction and allowing for details as needed.
- *Expressive power:* OWL employs description logic in a dynamic environment utilizing the open world assumption. Description logic is a powerful mechanism for stating concepts.
- *Standards:* OWL is based on RDF, which is a standard that is becoming more popular with many tools for processing formats.
- *Translatability:* As a formal language with well-defined semantics, OWL can be translated into other implementation languages, especially RDF-based languages. The degree to which the translation preserves all of the ontology features depends on the target language and its supported features.
- *Formal semantics:* OWL has well-defined semantics for language constructs. The semantics capability is supported by *Reasoner* specifications that describe what a valid structure should be.
- *Flexibility:* OWL offers a wide range of constructs to model concepts and relationships. In most cases, the modeler provides multiple choices to model any concept. The selection of a particular construct is usually determined by the use-cases for the concept and the relationships to other concepts.

- *User base:* OWL enjoys a strong user base for OWL, especially using the modeler Protégé. There are also many conferences and user group meetings and on-line forums supporting the development of ontologies in OWL.
- *Availability:* The OWL specification is published in the public domain and tools for modeling in OWL are available for free (e.g., Protégé and CMAP Tools[6]).

The next step is to translate the model into actual system implementation. Two aspects need to be addressed in this step, namely: verification tools; and, code generation. Checking tools are needed to make sure that the ontology structure is consistent and remains consistent during system operation, after changes have been made. Code generation tools assist in taking a formal ontology consistently and repeatedly from a formal language to an implementation language. System implementation typically goes through multiple iterations that may require re-writing the basic model, or large sections of it. Utilizing code generation tools makes this task easier.

Integration of the Ontology with Existing Ontologies

It is often the case that an ontology is being developed as an extension of an existing ontology or to connect with an existing ontology. In such cases, integration with the existing ontology must be carefully considered.

- Existence of other ontologies that are relevant to this ontology.
- All assumptions have to be made explicit.
- Agreement has to be achieved regarding concepts and relationships.

Analysis and Evaluation

The ontology must be examined from a technical perspective, along with the associated software environment, and the documentation with respect to a frame of reference, which includes:

- Requirements specifications.
- Competency questions.
- Real world appropriateness.

The selection of the frame of reference and the evaluation criteria have to align with the purpose and requirements of the ontology. The semantic correctness of an ontology is crucial for the proper functioning of applications. In order to evaluate an ontology it is useful to employ a methodology that has two main components: structural analysis; and, domain knowledge analysis.

[6] IHMC CMAP Tools facilitates the construction, navigation, and sharing of knowledge models represented as concept maps (see: http://cmap.ihmc.us/download/)

Structural Analysis: This involves the analysis of the structure of concepts in terms of hierarchy (taxonomy) and in terms of the relationships among concepts. The main criteria for this analysis are:

- *Uniqueness of concepts (no redundancy):* Every relevant concept in the domain should be represented in a clear and concise manner within the model. Concepts that are similar or have some common properties with other concepts should be represented in relationship to the existing concepts, either in a class hierarchy or through other types of relationships such as "part-of". The ease with which existing concepts co-exist and new ones can be added is a key indicator of the model's elegance and sophistication.
- *No circular reference should exist at any level:* Circular references can occur when a parent class in a class hierarchy inherits from a child class at any level down the hierarchy. This circular reference may not be obvious if the child class is more than two levels down from the parent class. Circular references are problematic because they confuse the semantics of the two concepts (e.g., "… a jet plane is a kind-of aircraft" and "… an aircraft is a kind-of jet plane").
- *Levels of Abstraction:* Class hierarchies can have any number of levels, where every level introduces more details to the classes at that level. The choice to add many attributes to a class in one level of the hierarchy or to create many levels with few attributes at each level has implications on the semantics of the model and on the operational aspects of applications that use this model.
- *Complexity (number of concepts + number of relationships for each concept):* The complexity of an ontology plays an important role in its usability. Applications typically traverse a collection of related concepts to form a context for reasoning or decision making. The more complex the ontology, the more involved it becomes for the application (and for the application developer) to form the proper context.

Domain Knowledge Analysis: Focuses on the *purpose* of the ontology. Use-cases for the application identify its information needs and form the basis for assessing the ontology's completeness. The criteria for this analysis are:

- *Coverage of use-cases (completeness):* Application use-cases define the different ways the ontology will be used. All concepts that are referenced in the target use-cases must exist in the ontology in some form (either directly or inferred). Other concepts not explicitly mentioned in any use-case may exist in the ontology serving as extended specifications for further reasoning or increased scope.
- *Partitioning:* The arrangement of classes in a hierarchy, where features of subclasses do not overlap, forms a *disjoint decomposition* of classes. When subclasses represent all the possible classifications of a super class, then this is called *exhaustive decomposition*. In this case, any instance of the super class is also an instance of one of the subclasses. These two properties of

ontology partitioning (i.e., disjoint decomposition and exhaustive decomposition) place integrity constraints on the ontology and provide for tighter semantics, as well as a more powerfully expressive ontology that in turn leads to more straightforward reasoning capabilities.

- *Extensibility:* When the incorporation of additional concepts is required, perhaps due to the need to support additional use-cases, it should be possible to add these concepts without the need to re-structure the entire ontology. If engineered correctly, the incorporation of extended or entirely new concepts can be achieved in a fairly isolated manner without unduly impacting unrelated areas of the model or actual model users.
- *Documentation:* The intended meaning and the usage of each concept must be clearly documented, so that reasoning facilities can effectively and appropriately employ them.

Documentation of the Ontology

The development of software components that are based on an ontology relies on good documentation of the ontology and the availability of the documentation to all developers. The documentation must include:

- Purpose and intended use of the ontology.
- Assumptions made at every level about concepts and their relationships.
- Primitives used to express the definitions (i.e., meta-ontology).
- Relationship to existing ontologies.

Using a lexicon such as WordNet standardizes the definitions across multiple developer teams and across organizations, and reduces the chances for ambiguity in dealing with concepts that may have multiple word-senses. The choice of WordNet also offers the opportunity for other ontologies to integrate with the ontology under consideration, by examining the standard definition of its concepts and deciding on concept compatibility.

References

Barber K., A. Goel, D. Han, J. Kim, D. Lam, T. Liu, M. MacMahon, C. Martin and R. McKay (2003); 'Infrastructure for Design, Deployment and *Experimentation* of Distributed Agent-based Systems: The Requirements'; The Technologies, and an Example, Autonomous Agents and Multi-Agent Systems. Volume 7, No. 1-2 (pp 49-69).

Brown P. (2008); 'Implementing SOA: Total Architecture in Practice'; Addison-Wesley.

Erl T. (2008); 'SOA: Principles of Service Design'; Prentice Hall.

Fahad M. and M. Abdul Qadir (2008); 'A Framework for Ontology Evaluation'; Proceedings International Conference on Conceptual Structures (ICCS), Toulouse, France, July 7-11 (pp. 149-158).

Forgy C. (1982); 'Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem'; Artificial Intelligence, 19 (pp. 17–37).

Jennings N., K. Sycara and M. Wooldridge (1998); 'A Roadmap of Agent Research and Development'; Autonomous Agents and Multi-Agent Systems, Vol. 1 (pp. 7-38).

Lu Q. and V. Haarslev (2006); 'OntoKBEval: A Support Tool for DL-based Evaluation of OWL Ontologies'; Proceedings OWL: Experiences and Directions 2006, Athens, Georgia, November 10-11.

Pohl J. (2003); 'USTRANSCOM: Corporate Information-Centric Environment (CICE)'; Report prepared for USTRANSCOM TCJ6-A (available from: CDM Technologies Inc, 2975 McMillan Avenue (Suite 272), San Luis Obispo, California 93401), October.

Supekar K. (2005); 'A Peer-Review Approach for Ontology Evaluation'; Proceedings 8th International Protégé Conference, Madrid, Spain, July 18-21.

Wooldridge M. and N. Jennings (1995); 'Intelligent Agents: Theory and Practice'; The Knowledge Engineering Review, Vol. 10(2) (pp. 115-152).

Wooldridge M., N. Jennings and D. Kinny (1999); 'A Methodology for Agent-Oriented Aanalysis and Design'; Proceedings Third International Conference on Autonomous Agents (Agents-99), Seattle, Washington.

Conveyance Estimator Ontology: Conceptual Models and Object Models

Xiaoshan Pan, PhD

Senior Software Architect
CDM Technologies, Inc., San Luis Obispo, California, USA

Abstract

This paper proposes the construction of a Conceptual Model as a logical step prior to the preparation of the Object Model of an ontology to facilitate the design and development of software systems in which a high-level internal representation of context supports some intelligent capabilities. The intent of the Conceptual Model is to be expressive for human interpretation utilizing descriptions that are readily understood by laypersons, subject matter experts, and software developers who may be concerned with only a particular portion of the software system. The intent of the Object Model (as a subset of an ontology) is to be expressive for machine interpretation. The author argues that the existence of a Conceptual Model not only serves as an effective communication vehicle among the various stakeholders in a software development project, but also facilitates the development of the Object Model component of an ontology.

Keywords: Conceptual Model, Object Model, ontology, representation, software development, UML, Unified Modeling Language

Prolog

In the beginning, there was the World. And then humans came. Humans studied the things that they perceived as being in the World, which they called Ontology. Ontology helped humans communicate with each other, share and reuse knowledge. Later, humans developed computers and the software systems running on the computers that helped to solve problems about specific aspects of the World, called Domains. In the process, humans learned that a computer system needed to have a representation of a domain to develop and specify problem-solving strategies for the domain. Such a representation was called by different names depending on the technology employed, such as Variables and Data Structures. Popular contemporary terms include Object Model, or Ontology.

Introduction

In the context of the computer and information sciences, an ontology is the knowledge representation of a domain in the world, and its purpose is to enable "… knowledge sharing and reuse" (Gruber 2008). Practically speaking, an ontology or the knowledge about a domain comprises three elements:

1. the concepts (i.e., objects with attributes) used to describe the domain, which provides shared meanings to those working within the domain;

2. the relationships and constraints among the concepts; and

3. the behaviors or business rules typically describing scenarios about the interactions among the concepts in the domain.

Although an Object Model might be targeted as the vehicle of the knowledge representation for a domain when constructing a computer software system, this target has not yet been able to fully function as an ontology. The reason is simple but not obvious – while an Object Model is able to represent the concepts, the relationships among concepts, and the constraints among the concepts (to some extent), the technology has not reached maturity where it can fully describe the behaviors (i.e., the business rules) that relate to the concepts. In software development, business rules are typically implemented separately from the Object Model. For example, a system built with a three-tier architecture contains a Presentation Tier, an Information Tier, and a Logic Tier. Business rules are implemented in the Logic Tier (e.g., agents) and the Object Model resides in the Information Tier. Therefore, as a medium to represent knowledge, an Object Model alone is not sufficient. In other words, an Object Model is not equivalent to an ontology (Figure 1).

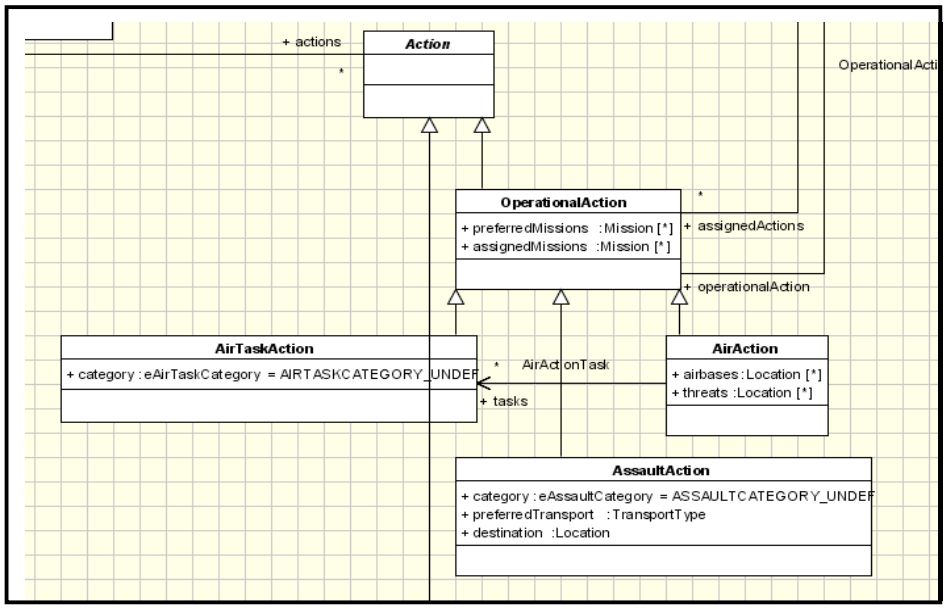


Figure 1: Part of an Object Model described in UML. Concepts and their relationships are shown, however it is not clear what knowledge it is intended to convey.

Another reason why Object Models and ontologies are not equivalent is that they can have different scopes. An ontology generally is built for capturing, sharing and reusing domain knowledge, and it can be applied to many areas as long as these areas have need for this knowledge. An Object Model is often a partial implementation of an ontology, meaning only a subset of the concepts and relationships are implemented. The choice of these concepts and

relationships is determined by the specific functional requirements and use-cases of a software system.

Since an Object Model is usually tied to a specific implementation and described in a formal language such as the Unified Modeling Language (UML), the words (or symbols) used in the model are most meaningful to the modeler himself, and can be difficult to interpret1 by other users (e.g., the users who might implement the business rules of the system). This is particularly the case when modeler's intentions and knowledge are not clearly documented and accessible to others. Therefore, as a medium providing shared meaning and understanding among human users, an Object Model is not an ideal choice2. Practice has shown that the more complex and expressive an Object Model is the more difficult it is to comprehend and thus to gain general acceptance.

The preceding discussion suggests that it is necessary to recognize the limitation of what an Object Model can represent functionally, and its insufficiency as an ontology. How then should an ontology be specified? The answer depends on who the ontology users are. If the users are software systems, then an ontology, in principle, can be represented by an Object Model, together with the business logic describing the behavior of the objects as defined in the model (possibly implemented as software agents). However, if it becomes difficult to retrieve and reuse knowledge provided in this way, then the purpose of an ontology is undermined. If the users are humans, including the clients who requested the system, domain experts, system architects, and component developers, a Conceptual Model may be more appropriate for presenting an ontology.

A Conceptual Model contains the domain concepts and their relationships described in plain English words. The relationships must be sufficiently flexible to describe not only normal UML relationships (e.g., inheritance, aggregation, composition, etc.) but also the behavior of the concepts as well. For example, Figure 2 shows a Conceptual Model about the Martian atmosphere. The model is composed of the following elements:

1. the concepts as used by the domain users, such as “Protective Ozone layer”, “Nitrogen”, and “Composition”;

2. the relationships among the concepts, such as “Composition includes Nitrogen, Argon, Carbon Dioxide”; and

3. the behavior of the concepts (i.e., describing the interactions among the concepts). For example, “Greenhouse gas absorbs some heat re-radiated by surface” and “General circulation of atmosphere leads to surface erosion”.

One can easily construct a Conceptual Model for the purpose of capturing, preserving, and sharing domain knowledge, which is precisely what an ontology is supposed to do. Therefore, it

---

[1] Complex and expressive object models can easily lead to different interpretations or misinterpretations.

[2] Note that this does not pose a problem to non-human users. An Object Model by far is still the most effective medium to provide shared understanding about a domain among multiple software systems.

seems reasonable to suggest the following propositions for the development of complex software systems3:

1. Subdivide the traditional development of an ontology into two distinctive processes: the development of a Conceptual Model; and, the development of a corresponding Object Model.

2. The Conceptual Model is best suited for capturing, preserving, and sharing domain knowledge among human users, while the Object Model is best suited for providing a shared representation about the domain in terms of objects and their relationships among different system components and/or different systems.

3. An Object Model is an implementation (or partial implementation) of its corresponding Conceptual Model in supporting some specific use-cases and system functionalities.

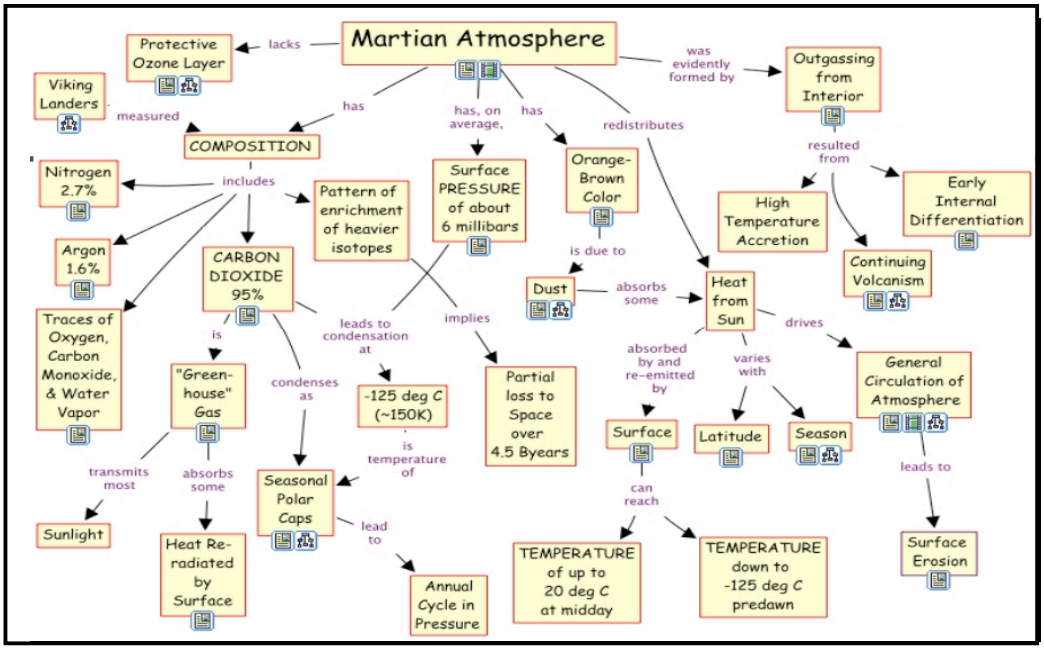


Figure 2: A Conceptual Model about the Martian atmosphere. The model serves well to capture the knowledge about the domain.

In this paper the author will refer to the development of the ontology for a Conveyance Estimator software system as a case study, to describe how the proposed idea was implemented in a specific project context. The objective of the Conveyance Estimator project is to provide capabilities for the assessment of logistical resource requirements—determining the approximate number of pallets, containers, aircraft, railcars, or trucks required to transport a given mix of supplies and equipment. At the micro level, the validity and accuracy of such estimation is largely dependent on the procedure used to optimally load a single container, or a conveyance,

3 These propositions may be necessary in developing all but the very simplest systems.

which is a complex problem in itself4. At the macro level, because different conveyance types represent different sets of constraints with significantly different cargo capacities (e.g., tens of thousands of items for a ship and less than a hundred items for a truck), it is reasonable for such an estimate to be made at both a fine-grained and a coarse-grained level.

Conceptual Model

The Conceptual Model for the Conveyance Estimator consists of a set of graph-like concept maps. Concept maps are graphical tools for organizing and representing knowledge (Novak and Canas 2008). The top level of the model is a "root" map (Figure 3), which defines a list of high level concepts and their relationships that shape the project, such as "Problem Statement", "Objectives", and "Conveyance Estimator Problem". Other concept maps are the expansions of these high level concepts.

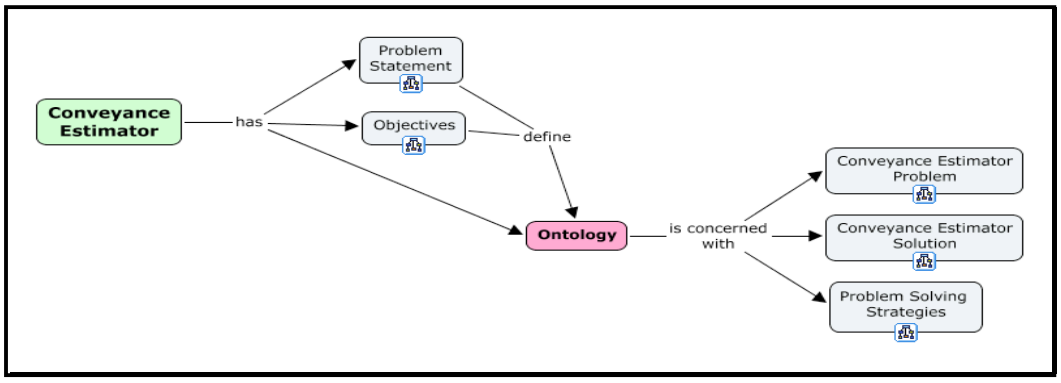


Figure 3: The top level concept map

The structure of this Conceptual Model is simple: each model is a collection of concept maps. A concept map is comprised of nodes and arrows. Both nodes and arrows can be attached to words or symbols. By convention, nodes represent concepts, and arrows represent relationships. Since verbs are freely used to describe relationships among concepts, such a graph can be easily used to describe the behavior of the concepts. Importantly, the model is presumably comprehensive to laypersons. For example, the top level concept map (Figure 3) can be read as "Conveyance Estimator has Problem Statement, Objectives, and Ontology", and "Ontology is concerned with Conveyance Estimator Problem, Conveyance Estimator Solution, and Problem solving Strategies."

Each concept within a concept map can be further expanded as another concept map. For example, the concept "Problem Statement of Conveyance Estimator" is further expanded as another concept map as shown in Figure 4. Similarly, any concept in Figure 4 can be expanded as well. For example, we could create concept maps to explain "equipment" and "supplies" (but that has not been done here since their meanings appear to be understandable.)

---

[4] Optimally stowing items into a three-dimensional container represents a bin packing problem, which is a combinatorial NP-hard problem within the realm of computational complexity theory.

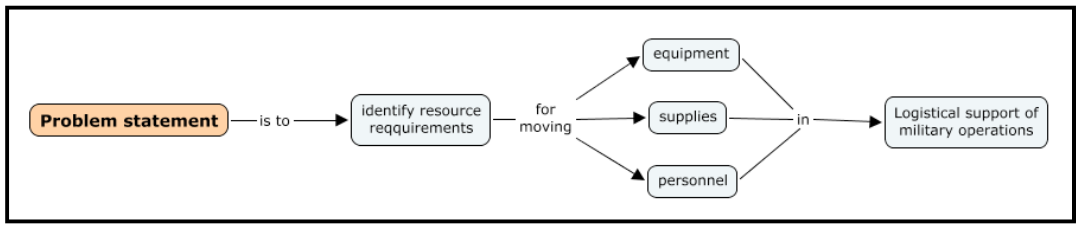


Figure 4: A concept map explaining the problem statement of the system.

Figure 5 contains a concept map describing the objectives of the system, which include "Web-based user access", "micro-assessment", "macro-assessment", and so on.

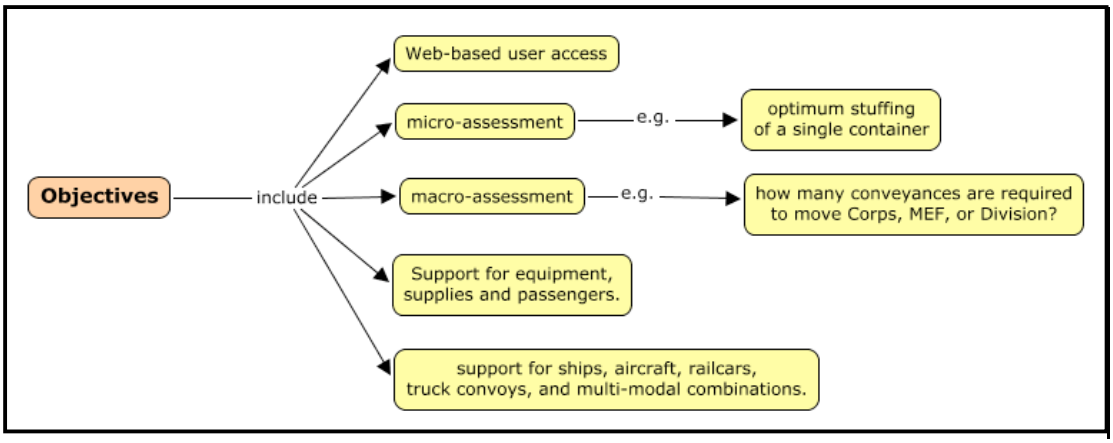


Figure 5: A concept map describing the objectives of the system.

Figure 6 describes the input of the Conveyance Estimator system. The input includes "requirement" and "planning options", and "requirement" is further defined as lists of "available conveyances" and "cargo items", and "conveyances" can be "pallet", "container", "truck", "railcar", "ship", and "aircraft", and so on.

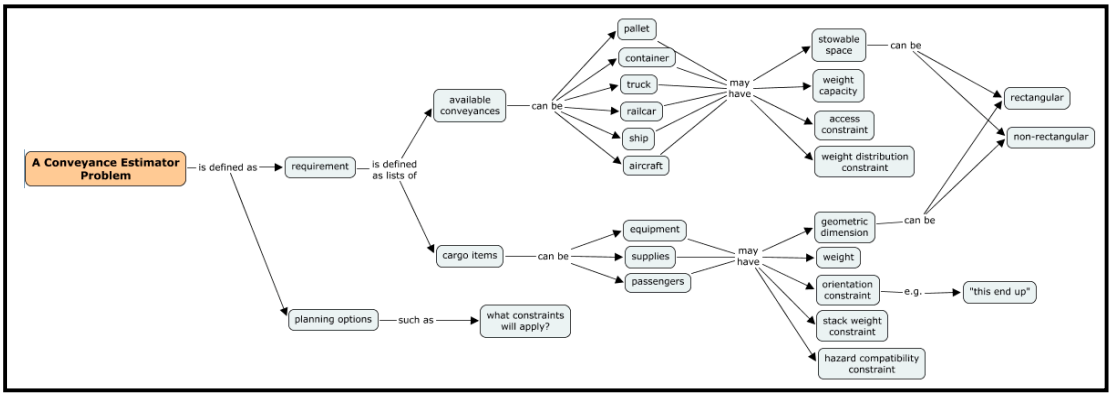


Figure 6: A concept map describing the system problem.

Figure 7 contains a concept map describing the kind of output that the Conveyance Estimator system produces. For example, one part of the output is a list of "packed conveyances" that holds the following information: "constraints applied to the package", "cargo items in the package", "packaging sequence", and "meta info".

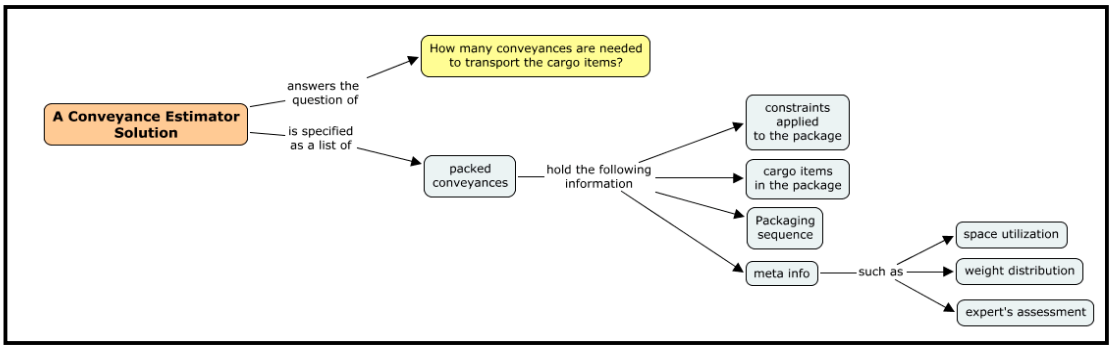


Figure 7: A concept map describing the system output

Figure 8 contains a concept map describing the possible "problem solving strategies" for utilization in the development of the Conveyance Estimator system, which include "computer simulation" and "case-cased reasoning".

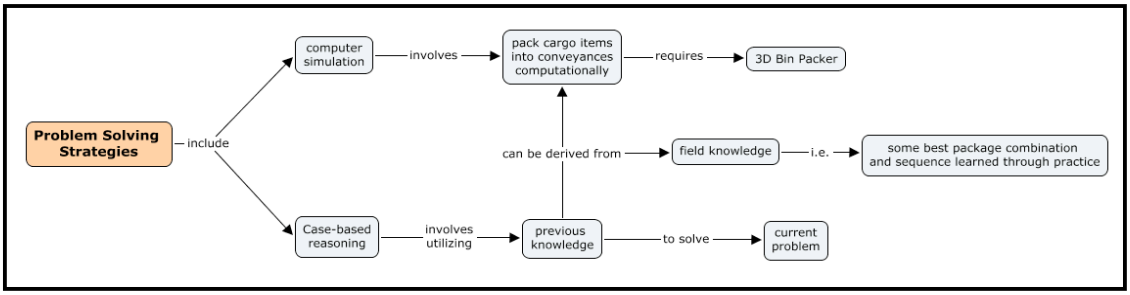


Figure 8: A concept map describing the problem solving strategies.

Object Model

By first looking at the Class Diagrams of the Conveyance Estimator (CE) Object Model5, the direct correspondence between them and the Conceptual Model as described in Section 2 may not be obvious for the following reasons:

1. While the Conceptual Model describes the high-level domain knowledge that the developers have gained through interaction with the customer, the domain experts, and the project advisory committee, the Object Model contains only the subset of concepts and relationships that are directly supportive of the limited functional requirements of the system. For example, in the concept map shown in Figure 6, an "available conveyance" is described as either "pallet",

[5] Please refer to the documentation of the CE Object Model at: \\datafish\jmadss\ConveyanceEstimator\doc\index.html

"container", "truck", "railcar", "ship", or "aircraft" type; however, in the Object Model as shown in Figure 10, there are only three types of bins (i.e., conveyances) – Pallet, Container, and Truck. This is because the developers decided that the first version of the Conveyance Estimator system would support only pallet, container and truck stowing. Therefore, adding other types such as "railcar" and "ship" was not necessary. Moreover, because the system will not initially support "railcar" stowing, adding a type "railcar" to the Object Model could confuse the developers (e.g., a GUI developer might create a report for "railcar" even though it is not supported by other components in the system).

2. The Conceptual Model and the Object Model follow different naming conventions. In the Conceptual Model, the words/sentences attached to each node and each arrow are plain English words/sentences. This is done for the sake of clarity, readability and ease of understanding. In the Object Model it is possible to use plain English words for some cases, but not for all cases6. For example, in the Conceptual Model as shown in Figure 6, an "available conveyance" has "stowable space", which can be either "rectangular" or "non-rectangular"; however, when that information is translated into the Object Model it becomes: a "BinType" (i.e., "available conveyance") has "widthLimit", "lengthLimit", and "heightLimit". This is because the modeler has decided that: (1) a "stowable space" will be treated as being "rectangular" in the system; and, (2) the width, length, and height of a "stowable space" will be called "withdLimit", "lengthLimit", and "heightLimit" in the model.

3. While the Conceptual Model is described in a non-constrained, very flexible manner, the Object Model must be described in a formal language such as UML. For example, an Object Model is organized in terms of "name space", "package", "class", "attribute", "data type", "generalization relation", "aggregation relation", and so on. Such formality is necessary so that the model descriptions are precise enough to be processed by software. However, there are no such restrictions in the Conceptual Model that is processed by human users.

4. While the Conceptual Model holds no other concerns than trying to describe the domain knowledge, the Object Model does involve some specific implementation concerns. For example, the package structure design of the CE Object Model (Figure 9) is largely based on how the system will be built and interfaced with other systems. For example, the CE Object Model has five first-level packages – "api", "entity", "type", "knowledgecase", and "security"; the packages "entity" and "type" are for holding the objects that exist in the domain; the package "api"7 is for holding the objects that will interface with other systems; and, the package "knowledgecase" is for holding the objects that implement the "Case-Based Reasoning" strategy.

---

6 In UML, one convention could be that white spaces are not allowed as part of a symbol.

7 "api" originally refers to Application Program Interface; here it is a placeholder for the objects that will be interfacing with other system, considering that CE system will be developed as a web-based service.

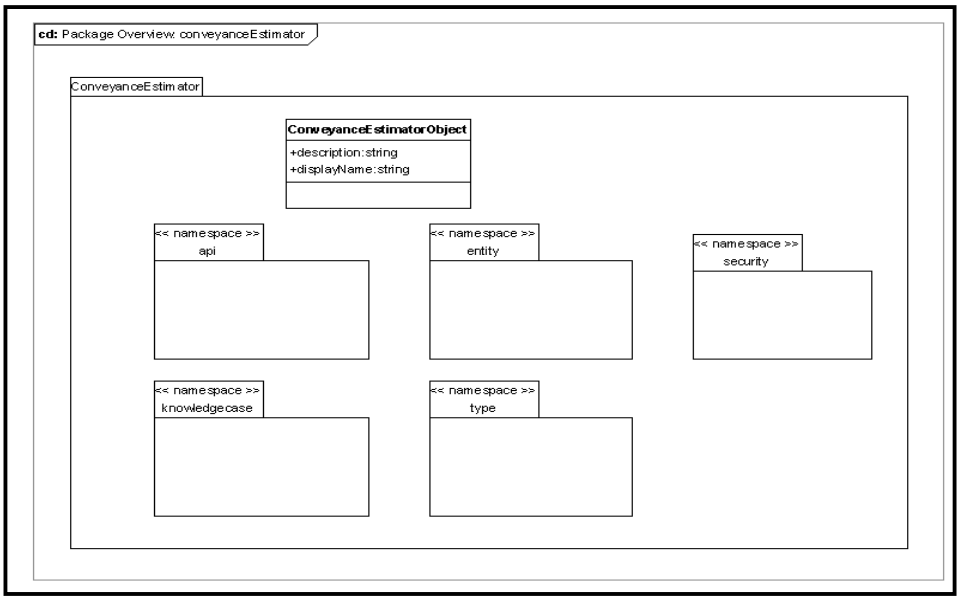


Figure 9: Top level package structure of the CE Object Model.

With all of the differences mentioned above, the two models are still very much coupled, due to the fact that the Object Model was developed utilizing the CE Conceptual Model as a guideline.

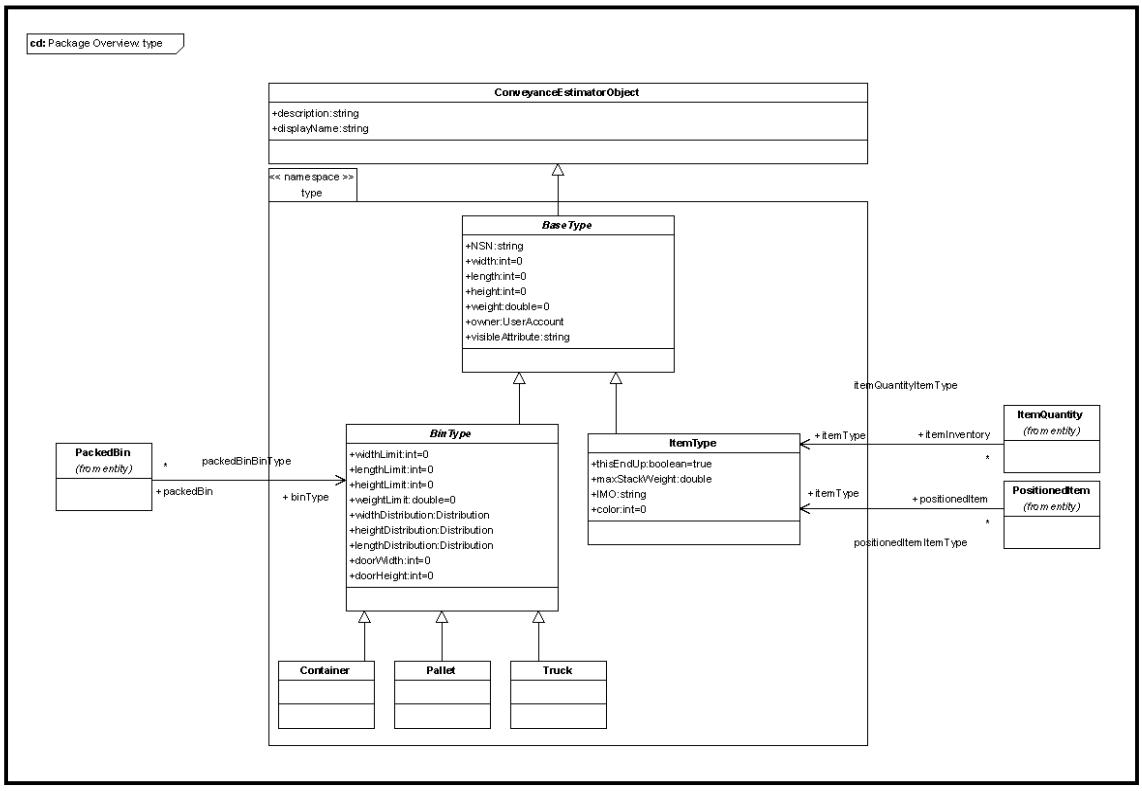


Figure 10: Package "type" of CE Object Model

Summary

Based on the experience gained in the development of the Conveyance Estimator Ontology, developing a distinct Conceptual Model prior to the development of an Object Model has shown to be beneficial in the following ways:

1. The Conceptual Model helps the modeler to capture (or describe) the domain knowledge more quickly and comprehensively.

2. During the development of the Conceptual Model, the modeler gains a better understanding of the domain concepts, the business rules, and the functional requirements, which will later help to clarify what needed to be built into the Object Model. Such clarity led to the smooth development of a clean and efficient Object Model for the CE system.

3. While the modeler can make the Conceptual Model as expressive as desired, the corresponding complexity is not necessarily reflected in the Object Model if it is not required to support the functional requirements of the system. Therefore, the Object Model users, such as component developers, do not have to deal with such complexity and can focus on component implementation in a straightforward way by working with a much simpler model.

4. The separation of the Conceptual Model from the Object Model appears to be a plausible solution to the on-going frustration between ontology developers and functional component developers. Such a separation makes it possible to satisfy the needs of both. The Object Model can be kept clean and efficient, with its evolution largely driven by the system's functional requirements and use-cases, while the Conceptual Model can grow as complex and expressive as needed to capture and preserve the domain knowledge.

5. A Conceptual Model serves as a means to understand its corresponding Object Model (especially in the case of complex and non-intuitive models), which directly facilitates the reusability of the Object Model across different applications.

There are many ways of utilizing a Conceptual Model in support of software project development. Because the general purpose of the model is for organizing and representing knowledge, depending on the types of knowledge it works with, a Conceptual Model can represent system architecture design, process flow, organization design, and many others aspects of a project, in addition to representing domain knowledge8. However, as far as the purpose of an ontology is concerned, in this writing the author has focused on its function in capturing, representing, and reusing domain knowledge.

---

[8] For example, the concept map as shown in Illustration 3 describes the high-level input and output of the CE system, which is a reflection of the design of the system architecture.

Furthermore, because it would be most useful if a Conceptual Model can automatically synchronize with its corresponding Object Model9, or vice versa, the next step in this research study will include exploring the possibility of auto-processing a Conceptual Model into a more structured and formalized model (even into an Object Model directly). A recent release of CmapTools (IHMC 2008) has support for building formalized ontologies utilizing concept maps. For example, CmapTools provides a set of predefined relation types for a user to choose from, such as "actives", "are", "type of", "at least", and "exact opposite of", and the semantics of these types can be defined in a machine processable format. In addition, the tool allows a concept map to be exported to and imported from standard XML files, which provides ready opportunities for tools to be created for processing a Conceptual Model automatically.

References

IHMC (2008). IHMC CmapTools. Available at: http://cmap.ihmc.us/

Gruber, T. (2008). "Ontology" Encyclopedia of Database Systems, Ling Liu and M. Tamer Özsu (Eds.), Springer-Verlag.

Novak, J. and Canas, A. (2008). The Theory Underlying Concept Maps and How to Construct and Use Them, Florida Institute for Human and Machine Cognition. Available at: http://cmap.ihmc.us/Publications/ResearchPapers/TheoryCmaps/TheoryUnderlyingConceptMaps.htm

[9] Currently the synchronization between a Conceptual Model and its corresponding Object is performed manually.

Increasing the Expressiveness of OWL Through Procedural Attachments

Dennis Taylor
CDM Technologies, Inc.
San Luis Obispo, California, USA

Abstract

The purpose of this paper is to provide an introduction to the OWL Web ontology language, a survey focused on the current state of the art in OWL inferencing capabilities, and a historical perspective on procedural attachments. The perspective is aligned with current OWL research. Several limitations of the OWL language and proposed extensions to overcome these limitations are discussed. A framework that provides empirical testing support for evaluating the effects of procedural attachments to the OWL inferencing capabilities is outlined. The examples presented suggest that it is possible to provide rule-based extensibility support for OWL that does not limit the ability of an OWL reasoner to perform consistency checking. Specifically, the framework is used to demonstrate with an example experiment the ability to provide support for the compound sub-property axiom.

Keywords: Google, inference engine, interoperability, ontology, OWL, OWL-DL, OWL-Full, OWL-Lite, OWL Reasoner, procedural attachment, World Wide Web.

1. The Semantic Web

One of the primary sources of information today is the World Wide Web, commonly referred to as the Web. It has revolutionized the way people communicate, exchange commodities, and even the way people think. The Web is accessed primarily by key-word searches through search engines such as Google, Yahoo, and MSN. Without these tools the Web may have never become the revolutionary technology that it is today. Key-word searches continue to improve, and give better access to the vast array of unstructured data on the Web. Regardless of this apparent success it has been recognized in the literature that key-word searches have several flaws. According to Parsons (2004) "… despite improvements in search engine technology, the difficulties remain essentially the same. It seems that the amount of Web content outpaces technological progress". Parsons identifies several difficulties that key-word searches have not overcome. For example, key-word search results are highly sensitive to vocabulary. Also, key-word searches deliver only sites that contain words included in the search query and the results are limited to a list of single Web pages.

These difficulties can be considered semantic problems. Wood (1985) defines semantics as the scientific study of the relationships between signs and symbols and what they denote or mean. With a deeper understanding of the semantics of words a search could return sites that contain synonymous terms in addition to the words searched for. Modern search engines also return only lists of single Web pages. They could be further enriched by providing an ability to link concepts from multiple pages. For the most part these problems are difficult to address due to the unstructured nature of the Web. It is difficult to determine appropriate semantics for data that are

unstructured. To properly determine the semantics of concepts in unstructured data most techniques require some defined context (Ceglowski and Cuadrado 2003). However, the majority of Web sites do not provide any context for the data found.

Some recent work has shown that it may be possible to gleam semantic information including context from unstructured data. Cilibrasi and Vitanyi (2004) have shown promising results by performing automatic meaning discovery using Google's search index. Advertisements displayed on Google's g-mail facility appear to exhibit some level of semantic behavior. For example, an e-mail message containing the term *thesis* will automatically trigger several advertisements for book binding services. Admittedly this apparently semantic behavior may be little more than a clever use of algorithms.

Although more and more clever algorithms may lead to richer semantic usage of the Web, a sizable community of researchers has begun to pursue a more proactive approach. They propose to organize data in a more structured manner in preference to attempting to extract semantic information from unstructured data. This approach, commonly referred to as the Semantic Web, provides a framework that allows data to be shared across multiple boundaries[1]. In this regard an understanding of the Semantic Web is critical to the value of this work.

2. The Problem of Interoperability

The fundamental aspects of the Semantic Web are external to the Web itself. They relate to a fundamental problem of Software Engineering in general, namely interoperability. It can be argued that the Web provides the first large scale example of how complex the problem of software integration really is.

Jackson (2005) describes the specifications of software as a boundary between the outside environment and the software. Adding a second software application to the environment shown in Figure 1 does not materially change the environment, if the two software applications are essentially isolated from each other (Figure 2). However, the situation is quite different if the two applications need to communicate and share data with each other.

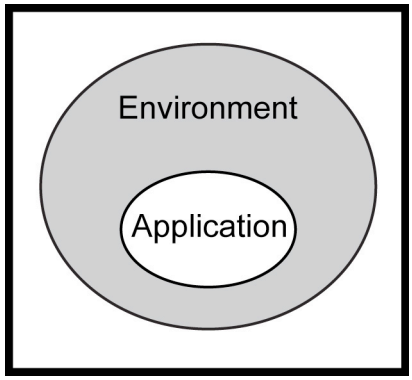


Figure 1: Single application

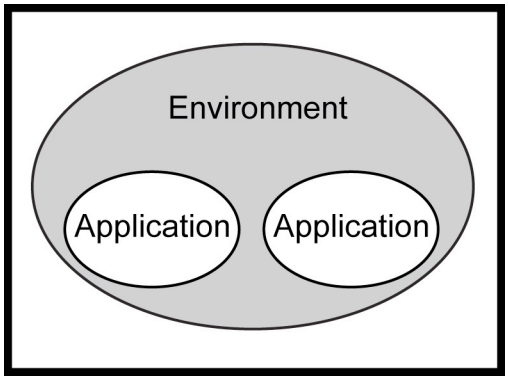


Figure 2: Two communicating applications

The need for interoperability between two or more applications has become an increasingly common requirement and has therefore become the subject matter of a great deal of research.

[1] See http://www.w3.org/2001/sw/.

Arguably much of the success of the Java programming language is due to its J2EE[2] interoperability capabilities provided by tools such as JBOSS, JMS, and SOAP. Using almost any standard programming technique results in clearly defined interfaces acting as a bridge or translator between the two systems (Figure 3). The complexity of a large number of interconnected systems can quickly become unmanageable (Figure 4).

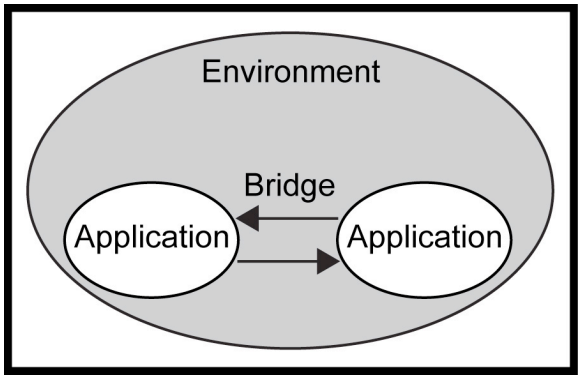


Figure 3: Manageable data exchange between two applications

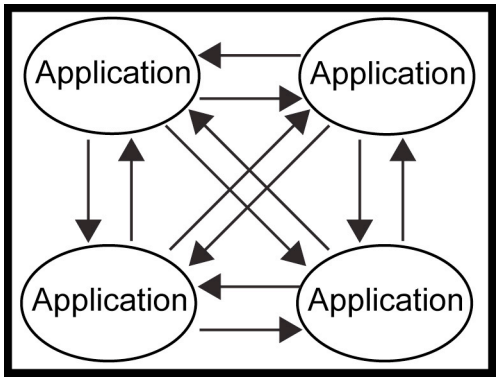


Figure 4: Unmanageable data exchange among four applications

To further illustrate this point imagine the following situation. If four prisoners speaking different languages were to be incarcerated in the same room it would make little sense for each prisoner to attempt to learn the languages of the other three prisoners.

3. Ontologies

According to Szolovitz and Ohno-Machado (2005) an ontology "… is a formal, explicit specification of a shared conceptualization. In other words, an ontology describes the concepts in the domain and the relationships that hold between these concepts. It is a shared vocabulary that can be used to model a domain (i.e., the objects and/or concepts that exist, their properties and relations".

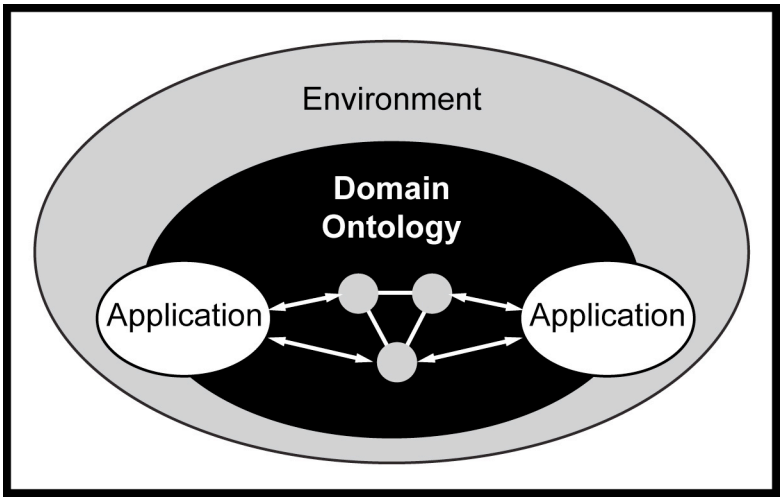


Figure 5: Two applications sharing the same semantics

[2] Java 2 Platform Enterprise Edition (J2EE).

If two software applications share the same semantics for the objects in their respective systems then they will be able to communicate within a meaningful context (Figure 5). Naturally, the entry of other applications into the environment shown in Figure 5 will require the addition of further concepts that must be described and shared within the domain ontology. Although the notion of an ontology is quite appealing there are several practical issues that need to be resolved. One of those issues is the need for a standard language to represent domain knowledge in the form of an ontology. The language recommended for this purpose by the W3C working group is OWL[3].

OWL has enjoyed a fair amount of interest since its introduction. One of the first large ontologies implemented in OWL was the GALEN biomedical terminology knowledge base (KB), which contains many biomedical concepts including anatomy, drugs, diseases, signs and symptoms (Rodgers and Rector 2006). The GALEN KB has been used in numerous applications in the medical field (Rogers and Rector 1996, Rosse et al. 1998, Wroe and Cimino 2001) and serves as one of the primary examples of the real world application of ontologies. The medical field incorporates a very large collection of shared information that is ideal for ontology representation. For instance the GALEN KB has been used to store diabetological terminology to assist in the diagnosis and treatment of Diabetes (Birkmann et al. 1997).

Originally developed under the direction of the Defense Advanced Research Projects Agency (DARPA) as an extension to the Resource Description Framework (RDF), the OWL ontology language has become widely recognized as a powerful means of representing knowledge.

4. OWL

The OWL Web Ontology Language is designed for use by applications that need to process the content of information instead of just presenting data to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with formal semantics. OWL has three increasingly-expressive sublanguages: OWL Lite; OWL DL; and, OWL Full.

OWL is a language specification that allows ontologies to be expressed in the form of a description logic. Of particular interest among the OWL documents provided by W3C is the OWL Use Cases and Requirements document found at http://www.w3.org/TR/webont-req/. This document outlines the essential requirements for OWL. The requirements specified are general enough to be good requirements for any appropriate ontology language. The first two requirements are essential for maintaining semantic clarity.

R1. Ontologies as distinct resources

R2. Unambiguous concept referencing with URIs

The first requirement addresses the need for every OWL ontology to be a unique reference. The second reiterates the importance of uniqueness by adding the need for unambiguous concept

[3] The OWL language is a revision of the DAML+OIL Web ontology language. DAML+OIL was developed by the "US/UK ad hoc Joint Working Group on Agent Markup Languages" that was jointly funded by the Defense Advanced Research Projects Agency (DARPA) in the US and the European Union (EU). The World Wide Web Consortium (W3C) created the Web Ontology Working Group, which published the first draft of the OWL language specification in July 2002. The OWL specifications became a formal W3C recommendation in February 2004.

references. In reference to URIs W3C states: "… Uniform Resource Identifiers (URIs, aka URLs) are short strings that identify resources in the Web: documents, images, downloadable files, services, electronic mailboxes, and other resources." OWL is ideal for the Semantic Web because every reference is a URI allowing any OWL document to reference any other by way of the Web. It is essential that the concepts in an ontology remain unambiguous, since a software program will not be able to distinguish one concept from another if there are ambiguities.

4.1 OWL Basics

While it is not intended to describe the syntax of OWL in detail, some of the basic concepts will be presented to facilitate further discussion. Since OWL is an extension of RDF it inherits the triple notation used by RDF. This triple notation is quite simple consisting of a subject, object, and predicate (Figure 6).

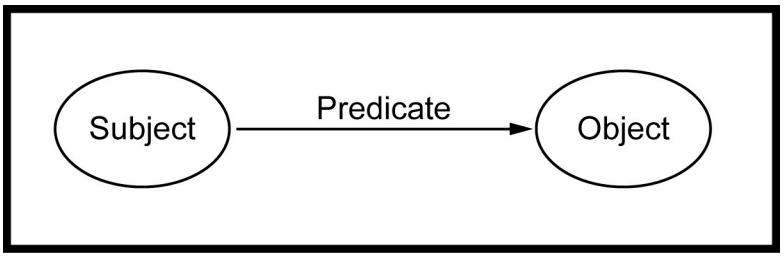


Figure 6: RDF triple notation

Anything expressed in an OWL ontology uses the triple graph notation of RDF. The fundamentals of OWL can be subdivided into: classes; data types; properties; property restrictions; and, individuals. Instead of adhering to the RDF naming convention for a triple of subject, predicate and object, OWL has chosen a more fitting terminology. The subject of a triple is always a class. The object of a triple can be a class or data type and the predicate of a triple is a property. A property can either be an object property (Figure 7) that links two classes or a data type property (Figure 8) that links a class to a data type.

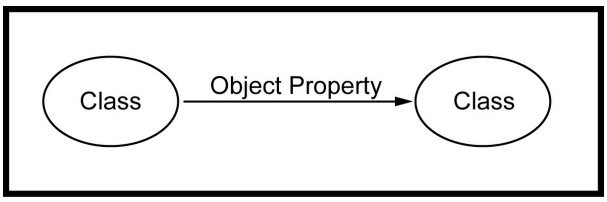


Figure 7: Object property

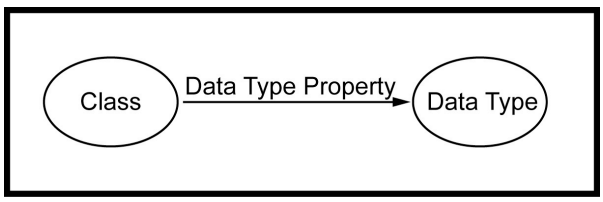


Figure 8: Data type property

Properties may also contain restrictions. One type of restriction is a value restriction. Such a restriction can limit the domain or range of a property. A restriction of a properties range will limit the classes or data types that can be used as the range for that property when applied to a particular class description. For example a *Human Child* class could have a restriction on the property *hasParent* to restrict the range of the property to be *Human*. In this way a *Human Child* could never have a *Tree* for a parent.

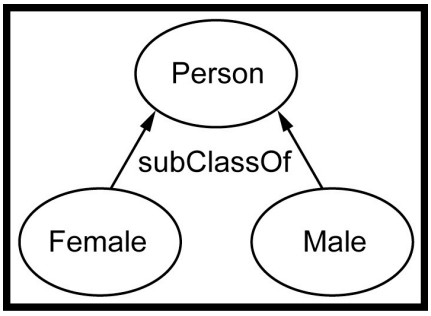


Figure 9: Family taxonomy

Another restriction type is the cardinality restriction that limits the number of values of a property that a class can have. Cardinality can be expressed as a minimum, maximum or both. For example a *Spider* would have both a maximum and a minimum cardinality of eight legs. An individual in OWL may be an instance of several classes or none of the classes defined. However, an individual will have a set of assertions that define its place in the concept hierarchy as well as its associations to other individuals. Once again these characteristics are all defined using the triple notation. By building a series of these triples with various restrictions a description of a concept can emerge. An ontology that contains some of the concepts that describe a family will be presented. This ontology has a fairly straightforward taxonomy (Figure 9) and will be used throughout the rest of this paper to discuss various features and limitations of OWL.

As shown in Figure 9, a person class is defined as the subclass of *owl:Thing*. The classes Male and Female are then defined as subclasses of the class Person. These are the only defined classes in this simple family taxonomy. For this ontology there are many more properties defined than classes. The following is a list of properties defined with their domain and range restrictions.

Property	***Domain***	***Range***
uncleOf	Male	Person
auntOf	Female	Person
nieceOf	Female	Person
newphewOf	Male	Person
siblingOf	Person	Person
brotherOf	Male	Person
sisterOf	Female	Person
parentOf	Person	Person
motherOf	Female	Person
fatherOf	Male	Person
childOf	Person	Person

The properties *brotherOf* and *sisterOf* are *subProperties* of *siblingOf*. The *motherOf* and *parentOf* properties are also the *subProperties* of the *parentOf property*. More detailed descriptions of OWL can be found in the OWL semantics and abstract syntax section of the W3C Web site.

4.2 OWL as an Ontological Medium

There are additional benefits for using OWL as an ontological medium. These added benefits derive from OWL's semantics and allow OWL reasoners to perform various inferencing functions, including but are not limited to:

- Checking Ontology Consistency to ensure that any instances of an ontology meet all of the restrictions of that ontology and do not produce any contradictions.
- Concept *satisfiability*, which checks if it is possible for a class to have any instances. If a class is *unsatisfiable*, then defining an instance of the class will cause the entire ontology to be inconsistent.
- Classification, which computes the subclass relationships between every named class to create the complete class hierarchy. The class hierarchy can be used to answer queries for all or only the direct subclasses of a class.
- Realization, which finds the most specific classes that an individual belongs to (i.e., computes the direct types for each of the individuals).

4.3 Consistency Checking

In systems like OWL where concepts are intended to be unambiguously defined it is very important that there is a way to maintain consistency of the information described. Inconsistencies in a stand-alone system can often pose problems but do not typically become an area of concern until interoperability between systems is necessary. Controlling inconsistencies in the Semantic Web is the primary goal of some OWL inference engines (Zou et al. 2004). They typically tackle the problem of eliminating inconsistencies in two stages. The focus of the first stage is to detect inconsistencies, while the second stage addresses the resolution of inconsistencies. One eventual goal of the Semantic Web is to enable trust of various systems. Once an inconsistency has been identified steps have to be taken to resolve the inconsistency.

For example, if one source states that a *Human* is the parent of a *Human Child* and another source asserts that there is a *Human Child* with a *Tree* as a parent then one of these sources must be wrong. It is obvious to a person that the second source is wrong since clearly trees do not have human children. Identifying which source is the most trustworthy may be an appropriate means of resolving the inconsistency. Some research has been conducted on the various uses of trust in the Semantic Web (Klyne 2002, Golbeck et al. 2003). In this paper we will be concerned only with the detection of inconsistencies and in particular the ability for an inference engine to perform consistency checking on an OWL document.

There are several inference capabilities of OWL reasoners that have received a great deal of attention. One of these is consistency checking. This capability is used to ensure that asserted facts in an OWL ontology are not contradictory. The simplest example of two contradictory statements is: A is true, and A is false. Since A cannot be both true and false, the statements are contradictory. While consistency checking is essentially involves the verification that an ontology has no contradictory statements, W3C provides the following more formal description

of consistency checking.

Definition: The translation of a separated OWL vocabulary, V' = VO + VC + VD + VI + VOP

+ VDP + VAP + VXP, written T(V'), consists of all the triples of the form
v rdf:type owl:Ontology . for v _ VO,
v rdf:type owl:Class . for v _ VC,
v rdf:type rdfs:Datatype . for v _ VD,
v rdf:type owl:Thing . for v _ VI,
v rdf:type owl:ObjectProperty . for v _ VOP,
v rdf:type owl:DatatypeProperty . for v _ VDP,
v rdf:type owl:AnnotationProperty . for v _ VAP, and
v rdf:type owl:OntologyProperty . for v _ VXP.

Definition: Let D be a datatype map. An Abstract OWL interpretation, I, with respect to D with vocabulary consisting of VL, VC, VD, VI, VDP, VIP, VAP, VO, satisfies an OWL ontology, O, if:

1. each URI reference in O used as a class ID (datatype ID, individual ID, data-valued property ID, individual-valued property ID, annotation property ID, annotation ID, ontology ID), belongs to VC (VD, VI, VDP, VIP, VAP, VO, respectively);
2. each literal in O belongs to VL;
3. I satisfies each directive in O, except for Ontology Annotations;
4. there is some o _ R with <o,S(owl:Ontology)> _ ER(rdf:type) such that for each Ontology Annotation of the form Annotation(p v), <o,S(v)> _ ER(p) and that if O has name n, then S(n) = o; and
5. I satisfies each ontology mentioned in an owl:imports annotation directive of O.

Definition: A collection of abstract OWL ontologies and axioms and facts is consistent with respect to datatype map D if there is some interpretation I with respect to D such that I satisfies each ontology and axiom and fact in the collection.

If this definition seems confusing then it may be replaced by a simpler alternative definition that is based on the formalism of the description logic, as follows.

An OWL ontology can be divided into two distinct parts, namely the terminological box (Tbox) and the assertional box (Abox). The TBox contains hierarchical concept definitions, whereas the Abox contains assertions that state where in the hierarchy individuals belong, and the associations between those individuals.

(1) Every employee is a person …….. belongs in the TBox, while the statement:

(2) Bob is an employee …….. belongs in the ABox.

With this distinction a better definition of Consistency Checking can be given as: the operation to check the consistency of an ABox with respect to a Tbox. If a Tbox defines a concept

Employee as in (1) above then if the individual *Bob* is asserted to be an *animal* in the Abox, the Abox will not be consistent with respect to the Tbox since *Bob* is an animal and not a person. The distinction between Tbox and Abox will become more important later in this paper.

The ability to perform consistency checking and the other inference capabilities mentioned previously depend on what parts of the OWL language are being used. Certain expressive forms of the language can lead to a loss in inference capabilities. The distinction between what leads to possible inference and what does not has prompted a split in the OWL language. Before discussing these distinctions it is important that the reader has a grasp of the concepts of *decidability* and *tractability*, since they are both fundamental in understanding the capabilities of OWL.

4.4 Decidable and Undecidable

Decidability has been an area of study since the humble beginnings of computer science when Alonzo Church and Alan Turing independently hypothesized about the nature of computable problems in what is commonly referred to as the Church-Turing thesis (Turing 1936, Church 1934). The thesis claims that any calculation that is possible can be performed by an algorithm running on a computer, provided that sufficient time and storage space are available. The most important result of this thesis is not so much what is computable but that there exist problems that are not computable and are therefore undecidable as far as a computer is concerned. This does not mean that it will take a long time for a computer to find the answer but that it is impossible for a computer to find an answer. One such problem is the Halting problem (Turing 1936). It is not the intention of this paper to provide a detailed account of this problem. Instead the reader is referred to the cited description or any book on computational complexity. However, it is very important to know when a problem is undecidable so that software developers do not waste their time trying to solve unsolvable problems. Later this will become important in understanding that some capabilities of OWL are undecidable.

4.5 Tractable

Another important concept in computer science and general problem solving is that of tractability. A tractable problem is one that is decidable within a reasonable amount of time and memory. Reasonable is a relative term but can be quantified to some degree: if a week or even a year is considered a reasonable period of time to answer a question than certainly a century would be an unreasonable period of time. Problems that fall into the category of intractable are those that are decidable but take an exponential amount of time to compute. As a problem becomes larger it takes an exponential amount of time longer to find the answer. The best way to comprehend this is to imagine the growth of an exponential problem. If an exponential problem has a size n equal to 20 then an exponential growth of n would be nearly a million. If each of those million possibilities takes one second then it will take about 12 days of computation time to solve the problem. Now consider a problem of size n equal to 40, only twice as large as the previous example. The time required to compute an answer for this problem is nearly 35,000 years. This would certainly be considered an unreasonable amount of time. Exponential problems are considered to be NP-Complete. Tractability will also become important in understanding some of the capabilities of OWL later in this paper.

4.6 Sub-Languages of OWL

The questions of decidability and tractability within OWL has lead to the distinctions of three sub-languages. Much of the ongoing academic work on OWL has been based on formalizing the capabilities of the language by proofs of decidability, presenting algorithms that are tractable, or proving that an algorithm is NP-Complete. It is readily seen that the OWL language in its most expressive form is highly undecidable and that the language in its most decidable form is not very expressive. The balance between expressiveness and decidability has created the necessary divisions in the language. This section will briefly describe the capabilities of each of the three sub-languages.

4.6.1 OWL-Lite

OWL-Lite is the least expressive of the sub-languages but maintains the most decidable and tractable capabilities. W3C has stated that "… OWL-Lite supports those users primarily needing a classification hierarchy and simple constraints." OWL-Lite maintains all of the restrictions of OWL-DL and adds several more. It is ideal for a simple class hierarchy because it allows classes to be subclasses of others. In the family ontology example a Male would be a subclass of Person. It also allows classes to be specified as equivalent to other classes. For example the Person class could be equivalent to a Human class.

OWL-Lite allows three basic restriction to be applied to classes. These include cardinality, allValuesFrom, and someValuesFrom. The cardinality restriction is limited to values of zero or one. A cardinality restriction of one would imply that a class has to have one value of the specified property. For example, the United States has one and only one president. A cardinality of zero implies that the class cannot have that property. A democracy cannot have a dictator. The United States would be considered a democracy since it does not have a dictator. The allValuesFrom and someValuesFrom restrictions provide a way to define a class from classical predicate logic and are equivalent to the classic (for-all) and (there exists) definitions. OWL-Lite makes an additional constraint on these restrictions that they must apply only to class names. A detailed description of these restrictions is not important for this work and is therefore omitted.

OWL-Lite allows properties to be transitive, symmetric, and inverses of other properties. When properties are defined with these restrictions an inference engine is able to automatically infer additional information about the individuals being processed. Examples of each of these characteristics were used to extend the Family ontology described previously.

Transitive – If Bill is the sibling of Mary and Mary is the sibling of Bob then
Bill is the sibling of Bob.

Symmetric – If Bill is the sibling of Bob then
Bob is also the sibling of Bill.

Inverse - If Mary is the mother of Susan then
Susan is the child of Mary.

The cardinality restrictions of OWL-Lite cannot be applied to properties that are also transitive. The restrictions provided by the semantics of OWL-Lite allow inference to be both decidable and tractable. Researchers at the University of Manchester have shown that OWL-Lite can be reduced to knowledge base satisfiability in the SHIF(D) description logic (Horrocks and Patel-Schneider 2004a). By showing that the capabilities of the various sub-languages are equivalent to certain description logic capabilities they have been able to tie OWL to the large body of work already produced in description logic.

4.6.2 OWL-DL

OWL-DL is a more expressive version of OWL-Lite. Every legal OWL Lite ontology is a legal OWL-DL ontology. Therefore, every valid OWL-Lite conclusion is also a valid OWL-DL conclusion. This is an important consequence. If part of an ontology may be classified as OWL-Lite then any inference that is possible on that part in isolation is still applicable to the ontology as a whole. Some inference problems can be performed on isolated parts of an ontology such as the classification of a single individual (Horrocks et al. 2000).

OWL-DL does not restrict the values specified for cardinality restrictions. However, it does maintain that properties that have cardinality restrictions are not transitive similarly to OWL-Lite. Although the constraints on OWL-DL may appear to be arbitrary, they are actually largely based on description logic reasoning capabilities. W3C states that "… in particular, the OWL-DL restrictions allow the maximal subset of OWL-Full against which current research can assure that a decidable reasoning procedure can exist for an OWL reasoner". The same group that has demonstrated that OWL-Lite can be reduced to knowledge base satisfiability in SHIF(D) description logic, has shown that OWL-DL can also be reduced in SHOIN(D), a more expressive description logic. Horrocks has published a survey outlining the state of art in description logic and current challenges that still exist ([Horrocks 2005).

4.6.3 OWL-Full

OWL-Full contains all of the OWL language constructs and provides free, unconstrained use of RDF constructs. It allows classes to be treated as individuals and this allows it to be very expressive. For example, an ontology in OWL-Full would allow a particular type of aircraft, say a C-17, to represent a Class of that object but also to be an instance of the AirplaneType class. OWL-Full actually takes this one step further so that all data values are also considered to be part of the individual domain. The use of OWL-Full negates the guarantees provided by OWL-Lite and OWL-DL. The language is largely undecidable.

4.7 OWL Inference Engines

Since the W3C recommendation of OWL there have been numerous attempts at creating an OWL inference engine. All of the inference engines examined take one of two approaches. The first approach is based on Description Logic reasoners and the Tableaux algorithm. The second approach uses restricted first order logic theorem proofs.

Table 1: Comparison of F-OWL and other OWL Inference Engine

	F-OWL	Racer	FaCT	Pellet	Hoolet	Surnia	Triple
Logic	Horn, Frame, Higher Order	Description Logic	DL	DL	Full FOL	Full FOL	Horn Logic
Support	OWL-Full	OWL-DL	OWL-DL	OWL-DL	OWL-DL	OWL-Full	RDF
Based on	XSB/Flora	Lisp	Lisp	Java	Vampire	Otter	XSB
XML Datatype	Yes	Yes	No	Yes	No	No	No
Decidable	No	Yes	Yes	Yes	No	No	Yes
Complete consistency checker	No	Yes (OWL-Lite)	Yes	Yes(OWL-Lite)	No	No	No
Interface	Java, GUI, Command Line	DIG, Java, GUI	DIG, Command Line	DIG, Java	Java	Python	Java
Query	Frame style, RDQL	Racer query language		RDQL			Horn logic style
Known Limitation	Poor scaling		No Abox support		Poor scaling	Poor scaling	Only support RDF

The research group at the University of Manchester (UK) has specified some improvements currently used for intractable problems that can be reduced to constraint satisfaction problems (Horrocks and Patel-Schneider 2004a). Many of these improvements are variations of a commonly used algorithm for OWL inference known as the Tableaux algorithm. These algorithms provide a reasonable solution to OWL-DL reasoning.

Table 1 (Zou et al. 2004) provides a listing of the most popular inference engines and their capabilities as of 2004. It is interesting to note that this is a very active field as the *Pellet* inference engine used for the experiments in this paper now supports most consistency checking in OWL-DL instead of the mentioned OWL-Lite.

Some more detailed information about each of these systems as of 2004 is provided by Zou (2004)]. The Racer system implements SHIQ(D) using a Tableaux algorithm. It supports OWL-DL and both Tbox and Abox reasoning, The FaCT system implements SHIQ, but supports only Tbox reasoning. Pellet implements SHIN(D) and includes a complete OWL-Lite consistency checker supporting both Abox and Tbox queries. Vampire is a FOL theorem prover (Riazanov 2003). Otter is also an FOL theorem prover and is used in the Surnia inference engine. There are also a group of inference engines that take advantage of a subset of FOL known as Horn Logic. These engines typically take advantage of already known Horn Logic tools such as Jena, Jess, Triple, and XSB. Higher order first order logic has been experimented with in some systems through the use of Flora (Zou et al. 2004).

4.8 Limitations of OWL

Several research groups have identified missing capabilities of OWL. One of the primary groups

is Horrocks' team at the University of Manchester. Several of their findings will be briefly presented in this section.

Complex Role Inclusion Axioms are not possible in any of the current sub-languages of OWL. These axiom take on two forms:

> ***ownership*** propagates from an aggregate to its parts (e.g., the owner of the car is also the owner of the car's parts;
>
> ***localisation*** propagates from a division to its aggregate (e.g., a trauma located in a part of a body structure is a trauma of the body structures (Horrocks and Patel-Schneider 2004a).

These are both valuable axioms that are not expressible in OWL. Work on the Galen medical terminology knowledge base has led to the need for expressing such axioms. The second axiom is very important so that trauma can be accurately diagnosed. If a patient has a trauma in a ventricle in his or her heart it would be useful if that trauma could be identified as a problem in the patient's heart. Or in the case of a torn cartilage meniscus in the knee the system should be able to identify a knee injury. These are critical distinctions since doctors typically specialize in areas of the body. For instance the patient with a heart trauma will need a cardiologist while the patient with a knee injury may need an orthopedic surgeon. It would be a unfortunate to have to explicitly define the doctor who works on every sub-part of every body part. Several attempts have been made to include these axioms in OWL. One such attempt used Grammar Logic (Baldoni and Martelli 1998), but has been proven to be undecidable. Horrocks' group has managed to show that with the RIQ description logic complex role inclusion axioms can be decidable. A benefit is that with some limitations of the SHIQ description logic it can be translated into RIQ. This restriction of SHIQ is not however identified within the W3C recommendation and therefore the distinction is not made. It is important to note that the satisfiability procedures for RIQ experiences enormous exponential growth, but has been validated on small examples within the Galen ontology.

Compound Sub-Property Axioms are also not currently possible in any of the sub-languages of OWL (Horrocks 2005). The compound sub-property axiom can be defined as the composition of several properties to assert another property. An example of this that will be used later in the experimental section of this paper shows that an uncle can be defined as the brother of a mother. Since the uncleOf property relies on the brotherOf property as well as the motherOf property it is not possible to express this in OWL.

Both the complex role inclusion axiom and the compound sub-property axiom discussed can be solved using other languages such as rule languages. This is a commonly recognized fact within the OWL and Semantic Web community. This limitation has lead to an apparent need for rule-based support that is somehow coupled with OWL. The logical foundations for the Semantic Web provide a diagrammatic view of the Semantic Web technologies and in this depiction they layer rules on top of the Ontological vocabulary in a segment labeled Logic (Figure 10). Since OWL provides both the Ontological vocabulary and some of the Logic there is a discrepancy between how rule-based support and OWL should be used in conjunction.

Several different approaches have been applied to add rule support to OWL. However, none of the approaches examined involves a completely extensible method such as the procedural attachment approach presented in this paper.

The Semantic Web Rule Llanguage (SWRL) is one of the most popular proposals for an OWL extension. SWRL combines OWL and *ruleml* and has been proposed by Horrocks and Patel-Schneider (2004b) to compensate for several of the lacking capabilities of OWL (Horrocks 2005). By exploiting the features of first order predicate logic in both description logic and rule languages this group believes that they will be able to extend the expressiveness of OWL-DL. The OWL inference engine Vampire was written as a general purpose first order logic theorem prover and has some support for SWRL (Tsarkov et al. 2004, Riazanov and Voronkov 2002).

Several other researchers, who also associated with the University of Manchester, have performed similar work and have provided a survey based on their perspective of the state of description logics (Hustadt and Motik 2005). They also discuss a combination of OWL-DL and rules, however, they define a subset of rules referred to as DL-safe to maintain decidability.

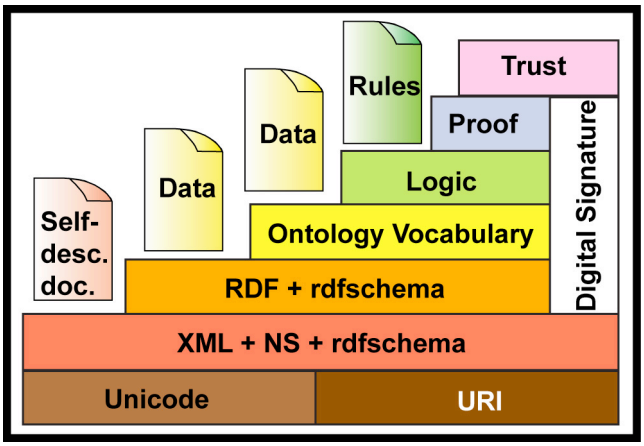


Figure 10: Logical foundations of the Semantic Web

There are a significant number of attempts to modify and extend the current OWL implementation to make it more expressive. Some examples include OWL-FA (Pan et al. 2005a), which is a language that lies somewhere between OWL-DL and OWL-Full and provides the added benefit of being able to represent hyponymOf relationships. This could allow reasoning on ontologies such as Wordnet in an OWL-DL context, which was previously not possible. Another proposed extension is f-SWRL (Pan et al. 2005b, Stoilos et al. 2005), a fuzzy extension of SWRL, and OWL-Eu (Pan et al. 2005c), which adds customization of data types to OWL. f-SWRL is an extension to SWRL that supports rules with fuzzy logic.

Some extensions do not propose new languages entirely but instead attempt to add one or more new capabilities. A position paper from an OWL workshop in 2005 proposes one built-in extension to OWL, namely the maximalSubPropertyOf (Pokraev and Brussee 2005). The number of proposed extensions to OWL is quite large. If OWL is going to reach a critical user base mass it is important that it maintains some degree of stability. The author of this paper believes that this stability may be better found in extending OWL instead of modifying it.

4.9 Procedural Attachment

The limitations and subsequent extensions of OWL are well grounded in mathematical logic such as first order logic, grammar logic, and description logic. Since the semantics and inference engines of OWL share a great deal of similarity with these types of logic, there is a natural tendency to extend OWL to support more of the axioms of these logic systems. The problem

with this approach is two-fold. The W3C recommendation for OWL does not indicate that OWL should become a general purpose logic language. The recommendation does suggest that OWL should be a description language that allows concepts to be described unambiguously. As such OWL is currently well suited to provide the capabilities necessary to meet this requirement. The description logic functionality provided by the OWL-DL sub-language is exactly that. Every extension carries with it a trade-off in language capabilities. The other problem with modifying the current semantics of OWL is that it is already a W3C recommendation to the contrary. This recommendation does not mean that a specification should not change but it does imply a resistance to change. The limitations of OWL should not be ignored and should be addressed by non-obtrusive approaches if possible. This section will provide a brief history of procedural attachments and provide an argument for the use of procedural attachments as a non-obtrusive extensibility feature for OWL.

Procedural attachment is not a new concept. As automatic theorem-provers and the initial expert systems were beginning to take shape some researchers saw an immediate need for procedural attachment support: "*… Consider the integers. Integers are concepts. Because there are infinitely many integers we need an infinite number of concepts. And because each concept needs to be defined, we would get an infinite number of frames! This is of course impossible in a concrete system, so some other way must be found to work with infinite sets of concepts. In particular we need mechanisms, i.e. Procedures, that will generate concepts as needed. Concepts like integers will be called infinitary concepts. Infinitary concepts are the first reason why we need procedural attachment*" (Steels 1979).

The problems associated with infinitary concepts are still prevalent today and are addressed in a procedural manner. For example,it would be impossible to describe all of the integers in an OWL ontology using separate classes for each. This problem is addressed in OWL using datatypes. Integers, and other data types are so common that OWL deals with them as separate entities. For example, an integer data type is used to represent the integers. A value can then be one particular integer without mandating that all integers are expressed as concepts. OWL inference engines are then able to use them in the same procedural fashion as they are used in the implementing language. This still does not entirely address the problem but makes it much easier to use common data types. What if some less common infinitary concept was needed such as the fibonacci sequence or the digits of PI? The most intuitive representation for infinitary concepts is provided by using procedural attachment.

There is another strong argument for procedural attachment that can be found in the same seminal paper from which the previous quotation was taken: "*… Here is the second reason. In certain situations we want to create a 'block box' that does a particular task in a way which is non-transparent for the reasoner. A typcial example of such a task is arithmetic. In principle it is possible to feed a particular axiomatization of number theory to the reasoner and perform arithmetic operations by performing deductions. It is also possible to create a system based on tables and methods mimicking the way human do arithmetic. But these approaches are only appropriate when we want to study axiomatic number systems or the way humans perform arithmetic. In most cases we just want the number crunching to be done by a 'black box' as efficiently as possible. The way to create such a black box is to use a programming language as the device to construct boxes with. A particular box is then invoked by associating a procedure with an aspect in a frame. Each time an instantiation of the frame is created an instance of the box is put at work. Another example where such a black box approach might be useful is in*

dealing with 'low level' tasks such as vision or sensori-motor behavior. The black box in this case is a channel that returns a piece of information or a mechanism that performs a possibly complex action" (Steels 1979).

The *black box* approach to hide the details has become one of the most prominent features in modern programming languages. The concept is used heavily in Object-Oriented (OO) programming and analysis. Procedural attachment in OO programming is defined as the process associated with an object to perform certain housekeeping operations. Any programmer who has spent much time with an OO language would be very familiar with such concepts. Many other logical systems have used procedural attachment in the past. One more recent example is the SNARK system developed at SRI International.

> "*… SNARK (Stickel) is a first-order-logic theorem prover with equality, associative commutative unification, and sorts. It contains a variety of standard machine inference rules, including resolution, paramodulation, and rewriting. It has a procedural-attachment mechanism, which allows certain computations, such as arithmetic operations, to be performed procedurally rather than axiomatically; it would be cumbersome to use the axioms of arithmetic to compute 22+22, when the computer already has built-in procedures for that purpose. We have seen that, for some areas such as temporal reasoning, it is advantageous to use special-purpose inference procedures rather than to rely on only SNARK's general-purpose inference rules. It is impossible, however, for SNARK to include every special-purpose procedure that may be useful for some application. Instead, SNARK includes a procedural attachment facility that allows a user to invoke external procedures as part of the standard resolution*" (Waldinger 2005).

SRI International is the largest artificial intelligence research center in the world. SNARK is a general purpose first order logic theorem-prover as stated. The fact that these provers are often used as the inference engines for OWL ontologies places procedural attachment only a short distance from OWL inference. For example, SRI has recently contributed a large amount of work to the area of ontology development and OWL. They have been instrumental in verifying the correctness of the axiomatic specifications of OWL provided by W3C. They used SNARK in their verification of the axiomatic specifications to provide proofs of correctness for the various axioms defined in OWL (Waldinger 2005).

After finding such close encounters between procedural attachment and OWL the author of this paper has been surprised to see very little evidence of procedural attachment being used in widespread OWL inference. The W3C recommendation for OWL explicitly defines procedural attachment as an objective, as follows:

> *013. Procedural Attachment: The language should support the use of executable code to evaluate complex criteria. Procedural attachments greatly extend the expressivity of the language, but are not well-suited to formal semantics. A procedural attachment mechanism for web ontologies should specify how to locate and execute the procedure. One potential candidate language would be Java, which is already well-suited to intra-platform sharing on the Web. http://www.w3.org/TR/webont-req/*

Regardless of this statement the majority of the literature that mentions procedural attachment claims that it is not supported by OWL. A position paper from the Workshop on Semantic Web Enabled Software Engineering (2005) makes the following claim:

"*… Accessing ontologies at run time as dynamic models has the advantage that generic algorithms like reasoners can be executed easily. However, such generic classes are on a different level of abstraction than the rest of the code, and for example don't allow for procedural attachment. I think what have in front of us is not only an extension of Model-Driven Architecture. We are in fact talking about a different development paradigm, which is attractive to many domains in the Semantic Web and beyond. The links into MDA are important and valuable on their own, but I think it's time to look into how to connect all this with other mainstream technologies*" (Knublauch 2005).

Although this is only a position paper it seems to point to the general sentiment held by most of the OWL research community. There is a stigma surrounding OWL inference that labels it as a different development paradigm. Description logics and the Semantic Web hold great promise, but the complete abolishment of procedural attachment has yet to be seen. The last comment from the above quotation is certainly accurate. There is a strong need to connect ontology capabilities to other mainstream technologies. The problem with this quotation is the apparent disparity between the W3C recommendation objective of OWL and the lack of support displayed for procedural attachment. After all one of the most mainstream technologies today is procedural programming.

Since the beginning of OWL researchers have been trying to identify its place. The medical field has viewed it as a place to catalog large collections of knowledge such as the Galen ontology. Others have seen OWL as a description layer between services. OWL-S is an extension of OWL that has added support for common Web service capabilities such as pre-conditions and post-conditions (Bryson et al. 2002, Martin et al. 2004). OWL-S is used to describe the capabilities and ways of accessing a service. A software program can discover a Web service and analyze its OWL-S service description. If the OWL-S document and the software program share the same meaning of the defined concepts then interaction can be automated. Another service-oriented approach has begun similar work from a different perspective: "*… We are interested in the faithful description of the changes to the world induced by the invocation of services. Services are thus actions that have pre-conditions and post-conditions. These conditions are expressed with the help of description logic assertions, and the current state of the world is described using a set of such assertions (A so called Abox) …*" (Baader et al. 2005a).

The difference between these two approaches is that the latter addresses a need to maintain some state of the world. The OWL-S approach describes the service and does not try and maintain the state of intermediate concepts. Maintaining the state of the world as OWL instances in an Abox allows each service to operate on the same instances, as services in its domain. This paper (Baader et al. 2005a and 2005b) addresses a need to formalize the changes that an action can make in the state of an ontology. It shows a translation from the Abox and Tbox formalisms to the situation calculus. Therefore, the proposed solution of the frame problem is identical to that originally proposed by Reiter (1991). By tying description logic to situation calculus OWL can also leverage the extensive work related to situation calculus. However, the paper does not address the architectural challenges of such an approach. For instance it would be difficult to manage the situation when a service is responsible for what changes. The order in which the changes have occurred would need to be defined to prevent every service from modifying the Abox at the same time. This approach is also only decidable with the implied restriction of OWL to disallow recursive concepts (Baader et al. 2005b). Procedural attachments are quite similar to services since they both provide some functional capabilities to the ontology. Procedural

attachments would just be closer to the inference engine and would not span the Semantic Web. If procedural attachments could be described in the form of pre-conditions and post-conditions a majority of the existing work on services could be incorporated into this work. This paper does not assume that this is the case and therefore leaves this as future work to determine if usefully procedural attachments could be fully described as pre-conditions and post-conditions of the ontology.

Until recently the majority of the use of OWL has been limited to Tbox reasoning with an empty Abox. Inference engines were written to only consider Tbox reasoning. However, in the last few years greater importance has been placed on Abox reasoning (Horrocks et al. 2000). The difference between a Tbox and Abox is reiterated here in a different way to provide another look at the definition. Horrocks gives the following definition of the distinction between Tbox and Abox: "*… the Tbox asserts facts about concepts (sets of objects) and roles (binary relations), usually in the form of inclusion axioms, while the Abox asserts facts about individuals (single objects), usually in the form of instantiation axioms*" (Horrocks et al. 2000).

The problem of determining consistency of a Tbox has been shown to be reducible to determining the satisfiability of a single concept (Horrocks et al. 2000). The common approach to this is by using a Tableaux decidability algorithm. Many applications of OWL have little use for Abox reasoning. Reasoning in most description logics, including OWL-DL, can be quite computationally expensive and including an Abox with many entities would certainly compound the issue. There are however valuable uses for the inclusion of Abox reasoning. One is the inclusion of state in an ontology. A Tableaux based algorithm has been created for deciding the satisfiability of unrestricted OWL-DL that extends the existing consistency algorithm for Tboxes (Horrocks et al. 2000).

Not all actions that can be performed on an Abox can be described in terms of post-conditions. It is not possible to define the results of an indeterminate action until the action actually takes place. The effects of an action may also be determinate at one level of description but indeterminate at another. For instance an ant's motion can be described as chaotic in which case it is indeterminate. The ant's motion could also be described in more detail to account for its ability to sense and follow pheromones. This more detailed description may remove some of the indeterminism in the ant's motion. The need may arise to address certain concepts in an OWL ontology at a high level due to inaccurate or incomplete data. Even if every concept is well understood there exist indeterminate actions that use random values, which are impossible to determine until the action has been completed. A service framework that does not account for indeterminate actions similarly to the way it accounts for determinate actions will have difficulties in coping with change if indeterminate actions are deemed necessary. Instead of specifying the effects of an indeterminate action the things that are guaranteed not to be affected can be stated. This poses a slightly different problem. If for every action a list of unaffected items is provided these lists could be very large. The problem of how to reason about change without needing to specify frame axioms for everything that does not change has become known as the frame problem. In 1969 McCarthy and Hayes first mentioned the frame problem (McCarthy and Haynes 1969). When examining the effects of procedural attachment it will become important to identify the pieces of an ontology that should not be changed in order to maintain inferencing capabilities such as consistency checking.

There has been some work using procedural attachments with OWL inference. In 2002 an inference system was written for analogical processing on an analogy ontology.

> "*… The semantics of the ontology are enforced via procedural attachment, using cognitive simulations of structure-mapping to provide analogical processing services... Just as predicates involving arithmetic typically have their semantics enforced via code, we use procedural attachment to enforce the semantics of the analogy ontology. This is crucial because special-purpose software is needed for efficient large-scale analogical processing, where descriptions involving hundreds to thousands of relational propositions are matched. These procedural attachments provide the means for analogical processing software to be seamlessly used during first-principles reasoning. .Procedural attachment requires two-way communication between the reasoning system and the attached software. The reasoning system must have some means for recognizing predicates with procedural attachments and carrying out the appropriate procedures when queries involving them are made*" (Forbus et al. 2002).

At first glance the procedural attachment approach used here does not seam to provide any new functionality. It simply validates the semantics of the ontology using procedural attachments. Essentially one could say all OWL inference engines do this. Under further examination it becomes apparent that they are actually performing sophisticated reasoning using procedural attachments. The analogies in their systems are not fully described within the OWL document because OWL does not support the expressiveness that would be required. They instead define the terminology of the analogies in OWL and then provide the reasoning through external algorithms (Forbus et al. 2002).

5. Contributions of this Work

In order to best analyze the effects of the procedural attachment approach on OWL inferencing the problem needs to be examined from two different angles. One way is to examine all possible modifications that can be made to an OWL ontology and provide proofs to identify their impacts. The second is presented in this paper by providing a framework for empirical testing. Empirical testing by itself will not suffice to provide adequate analysis but does provide the ability to rapidly identify possible results. This is the approach taken in this paper.

In this section a framework for rapidly testing empirical results is presented that integrates with the *Pellet* OWL reasoner. Several example experiments are conducted within this framework to determine the effects of various procedural attachments. In particular, the effects these experiments have on consistency checking is presented. Then a distinction is made between procedural attachments that maintain the ability to perform consistency checking and those that cannot guarantee consistency checking and still remain decidable.

5.1 The Framework

There are two primary framework components. The first is used to identify how and when procedural calls are made from within the inference engine. The second identifies how the procedure will be allowed to modify the ontology.

Procedural calls are allowed to register a call back with the inference engine for a particular OWL ontology. This is accomplished by specifying a list of the properties the procedure is interested in. Any time one of these properties is added or removed from the ontology the procedure will be invoked and informed of the property change by an event. The two event types are either a removal or an addition of a property assertion. Since OWL is a derivative of RDF, as

mentioned earlier, it maintains the triple notation of subject, predicate, and object. This implies that the entire ontology is in this triple notation and everything in the ontology is there by asserting these triples. A procedural attachment can subscribe as a listener to any property change, which is the same as saying it can subscribe to any predicate change. This allows the attachments to be invoked by any possible change within the ontology.

Changes to the ontology by a procedure are applied by removing or adding data type or object type properties. This allows unrestricted modification of the ontology. It is important to note that any practical use of a procedural attachment should not allow such an unrestricted modification, since this would make it possible to change the ontology in such a way that it is no longer valid in OWL. For example, every OWL document maintains the assertion that owl:Thing is a subclass of owl:Class. If this assertion were to be removed then owl:Thing would no longer be properly defined and the ontology would no longer be a valid OWL ontology. The experiments in this paper attempt to find a middle ground between unrestricted modification and no modification.

5.2 Experiments

The framework presented in this paper was developed to facilitate the extension of OWL to support rules. A rule is typically applied by testing the antecedent of the rule and if it is true then applying the consequent action of that rule. In this case a procedure is simply a Java object that implements a standard interface. This interface has only one method, namely changeOccured that takes an OWLEvent object as an argument. The object contains two elements, an enumeration that indicates the type of change that occurred and a copy of the triple that was added or removed from the ontology. The listening object can then store some internal state about the truth values of its particular rule. The example that will be discussed here allows support for the compound sub-property axiom discussed earlier. The types of rules or procedures that are implemented by the listener object are not restricted to this axiom and could take many open-ended forms. The compound sub-property axiom was presented as a limitation of OWL and therefore makes an ideal candidate for this example.

Since the rule antecedent has a number of conditions that must be met it may take several changedOccured calls before any action will be made by the procedure. The goal of this example is to provide support for the compound sub-property axiom in general, while first using a simple classical example. In the example of the *family* ontology discussed earlier there were several defined properties including brotherOf and parentOf. These are the only two properties necessary to demonstrate how the listener object subscribes to changes of these two properties. This is accomplished by passing a list of property identifications (IDs) to the Abox listener in order to subscribe the listener object. The antecedent of this rule is met whenever a *Person* has a *brother* and is also a *parent*. This is monitored by making a list of every person that has a brother and a list of every person who is a parent. If these two lists contain the same person then that person's brother is in fact the uncle of that person's child. When the antecedent has been met then the procedure will add a new uncleOf triple to the ontology.

The initial *family* ontology was modified to contain several individuals. These individuals were added and associated with each other by the following triples.

Bill isA Male,

Mary isA Female,

Susan isA Female,
Bill brotherOf Mary,
Mary motherOf Susan.

The starting main class of the *Pellet* reasoner was modified to accept a file that contains a list of the listener object classes that should be loaded when starting the reasoner. It will then create an instance of each of these classes and these will register themselves with the *Pellet* Abox. For the *uncle* example it creates just the one listener object. Then the *Pellet* reasoner will load the ontology as normal. The Abox class was modified to check if the properties being asserted are properties with attached procedures and calls the appropriate listeners as necessary. As it loads the above individuals and their associations it makes two calls to the attached listener object. One indicating that Bill has been asserted as the brother of Mary, and the other to indicate that Mary has been asserted as the mother of Susan. On the second assertion it met the antecedent of its rule and therefore asserted in the Abox that Bill is the uncleOf Susan.

This example is quite simple but demonstrates that any compound sub-property axiom could be supported by the framework in a similar manner. Programmers can create any listener objects they desire and subscribe them to listen to any number of properties. Then by storing the internal state asserted properties and validating when the antecedent has been met it can perform the desired action on the ontology by asserting new triples.

After all of the individuals have been loaded into the *Pellet* reasoner it continues on its initial intended task. For the sake of these examples and this work only consistency checking was performed by the reasoner. Consistency checking is one of the most fundamental capabilities of an OWL reasoner but as mentioned earlier it is quite critical. In the above example the assertion of Bill as the uncle of Susan did not create any unexpected inconsistencies.

In order to further test the effect that procedurally asserted triples have on consistency checking another test was conceived to force an inconsistency. In this test another listener object that subscribed to the property isA was created. Any time this listener is called with the subject of the triple being the class *Male* it will assert a triple indicating that the same individual is also *Female*. Since the two classes *Male* and *Female* where asserted in the Tbox as disjoint the new assertions will create an inconsistency. Running this test proved that the *Pellet* reasoner is indeed able to identify the inconsistency and report it. So the reasoner was able to accurately perform consistency checking to identify consistent and inconsistent ontologies after the procedural attachment was executed. The changes described this far have only made Abox assertions and have not modified the conceptual properties of OWL. In both of the examples so far the *family* ontology remained an OWL-DL ontology after the changes.

Another experiment was conducted that created new Tbox assertions, forcing the *family* ontology into an OWL-Full ontology. An object listener was written to once again listen to the isA property. Then, any time the listener is called it will assert that the class is also an instance. This was accomplished through a rather pointless assertion that *Female* is an instance of *Female* (i.e., *Female* isA *Female*) and the same for *Male*. This created an individual as a Class and therefore made the ontology OWL-FULL. In this case, *Pellet* entered into an infinite loop and had to be stopped manually. At this time the author is still unsure exactly why the infinite loop occurred. While *Pellet* is able to answer Unknown for some OWL-Full consistency tests, it was not able to do so in this case.

A restriction was then made to the framework to prevent such occurrences. This restriction limited the assertions to only Abox assertions. By limiting the framework to listen to and modify only individual assertions on user defined properties all the tests run created an ontology that retained its original sub-language. After further investigation the discovery was made that it is in fact impossible to modify the sub-language of an ontology through Abox assertions only. This was discovered by examining the *Pellet* code. *Pellet* is able to identify the sub-language for a particular ontology. It does this by examining only the Tbox assertions and makes no calls to the Abox. Admittedly, examining only the source code of this one reasoner does not amount to a formal proof. A formal proof of this is beyond the scope of this paper but should be examined and is left to future work.

5.3 Review and Future Work

The framework provided here with or without restrictions proved itself as a good tool for running empirical tests. It has only been used here for consistency checking but could be used extensively to identify the effects of procedural attachments on the other forms of inference that OWL reasoners can perform. The initial results of this work show that procedural attachments are quite promising for maintaining ontology consistency checking abilities. It falls far short of identifying all of the effects of procedural attachments on OWL ontologies in general. A great deal of further work is required to accomplish that goal.

One of the primary inference capabilities of OWL reasoners is entailment. *Pellet* and other OWL reasoning engines have followed an approach suggested by Horrocks and Patel-Schneider (2004b). In this approach Horrocks used the similarities between OWL and description logic to show that OWL entailment can be reduced to knowledge base satisfiability in the SHOIN(D) and SHIF(D) description logics. Several other reasoners take this approach as well including Racer, which is a commercial grade OWL reasoner (Haarslev and Moller 2001). If the procedural attachment approach can be shown to maintain the ability to perform entailment under similar restriction then this work would be a valuable contribution to the OWL community. However, currently standing on its own, this works does not provide enough proof about the effects of procedural attachment to warrant widespread use. Nevertheless, it is the author's belief that with appropriate restrictions placed on procedural attachment it could be a valuable asset to OWL inference engines in the future.

5.4 Conclusions

This paper has presented an argument for the use of procedural attachment within OWL. It has verified that consistency checking of OWL ontologies remains possible if the assertions made by procedural attachments are restricted to Abox assertions only. The author readily admits that this restriction may not be absolutely necessary but feels that even with the restriction the expressiveness of OWL can be greatly increased. This has been demonstrated by the use of procedural attachments like the sub-property axiom example presented. Some of the reasons for allowing rule support in a language like SWRL, such as the ability to share rules in a Semantic Web environment may eventually prove more useful then the procedural attachment approach. The ability to attach infinitary concepts and 'black box' capabilities should not be ignored. These capabilities can increase the expressiveness of OWL as well. Recognizing when an existing language or software system can be extended through procedural attachments has been shown to be valuable in the past (e.g., first order logic theorem proving) and should be examined in more

detail in order to increase the expressiveness of OWL.

6. References

Baader F., Lutz C., Milicic M., Sattler U. and Wolter F. (2005a); 'A Description Logic Based Approach to Reasoning about Web Services'; Proceedings of the WWW 2005 Workshop on Web Service Semantics (WSS2005).

Baader F., Lutz C., Milicic M., Sattler U. and Wolter F. (2005b); 'Integrating Description Logics and Action Formalisms: First Results'; AAAI 2005: 572-577.

Baldoni L. and A. Martelli (1998); 'A Tableau Calculus for Multimodal Logics and Some (Un)decidability Results'; in H. de Swart (ed.) Proceedings TABLEAUX'98, Volume 1397 of LNAI (pp. 44-59).

Birkmann T., J. Diedrich and M. Engelbrecht (1997); 'A Formal Model of Diabetological Terminology and its Application for Data Entry'; in Proceedings Medical Informatics Europe '97, Thessaloniki, Greece, IOS Press Volume 43.

Bryson J., Martin D., McIlraith S. and L. Stein (2002); 'Semantic Web Services as Behavior-Oriented Agents'; IEEE Computer 35(11) (pp. 48-54).

Ceglowski A. and J. Cuadrado (2003); 'Semantic Search of Unstructured Data Using Contextual Network Graphs'; ??, June.

Church A. (1934); 'An Unsolvable Problem of Elementary Number Theory'; Proceedings National Academy of Science, US, 20 (pp. 584-590).

Cilibrasi R. and P. Vitanyi (2004); 'Automatic Meaning Discover Using Google'; ??.

Forbus K., T. Mostek and R. Ferguson (2002); 'An Analogy Ontology for Integrating Analogical Processing and First-Principles Reasoning'; 18th National Conference on Artificial Intelligence, ??.

Golbeck J., B. Parsia and J. Hendler (2003); 'Trust Networks on the Semantic Web'; Proceedings of Cooperative Intelligent Agents, Helsinki, Finland, August.

Haarslev V. and R. Moller (2001); 'Description of the RACER System and its Applications'; Proceedings of Description Logics.

Horrocks I. and P. Patel-Schneider (2004a); 'Reducing OWL Entailment to Description Logic Satisfiability'; Journal of Web Semantics, 1(4) (pp. 345-357).

Horrocks I., U. Sattler, S. Tessaris and S. Tobies (2000); 'How to Decide Query Containment Under Constraints Using a Description Logic'; Proceedings of the 7th International Conference on Logic for Programming and Automated Reasoning, LPAR-2000, Lecture Notes in Artificial Intelligence, Springer-Verlag.

Horrocks I. (2005); 'Applications of Description Logics: State of the Art and Research Challenges'; in F. Dau, M. Mugnier and G. Stumme (eds.) Proceedings of the 13th International Conference on Conceptual Structures, ICCS'05, Number 3596, Lecture Notes in Artificial Intelligence, Springer-Verlag (pp. 78-90).

Horrocks I and U. Sattler (2004); 'Decidability of SHIQ With Complex Role Inclusion Axioms';

Artificial Intelligence, 160(1-2), December (pp. 79-104).

Horrocks I., U. Sattler and S. Tobies (2000); 'Reasoning With Individuals for the Description Logic SHIQ'; in D. MacAllester (ed.) Proceedings of the 17th International Conference on Automated Deduction (CADE 2000)'; Number 1831, Lecture Notes in Artificial Intelligence, Springer-Verlag (pp. 482-496).

Horrocks I. and P. Patel-Schneider (2004b); 'A Proposal for an OWL Rules Language'; Proceedings of the 13th International World Wide Web Conference, WWW 2004, ACM (pp. 723-731).

Hustadt U. and B. Motik (2005); 'Description Logics and Disjunctive Datalog – The Story So Far'; Description Logics, ??.

Jackson M. (1995); 'Software Requirements and Specifications'; Addison-Wesley, ??.

Klyne G. (2002); 'Framework for Security and Trust Standards'; SWAD-Europe, December.

Knublauch H. (2005); 'Ramblings on Agile Methodologies and Ontology-Driven Software Development'; Workshop on Semantic Web Enabled Software Engineering (SWESE), 4th International Semantic Web Conference, ISWC-2005.

Martin D., M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, M. Solanki, N. Srinivasan and K. Sycara (2004); 'Bringing Semantics to Web Services: The OWL-S Approach'; Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition, SWSWPC-2004.

McCarthy J. and P. Hayes (1969); 'Some Philosophical Problems From the Standpoint of Artificial Intelligence'; in Meltzer B. and D. Mitchie (eds.) Machine Intelligence 4 (pp. 463-502).

Parsons S. (2004); 'The Semantic Web Primer by Grigoris Antoniou and Frank van Harmelen'; MIT Press, Cambridge, New York

Rector A. and J. Rogers (2006); 'Reconstruction of the GALEN Upper Ontology in OWL'; Department of Computer Science, University of Manchester, Manchester, England.

Pan J., I. Horrocks and G. Schreiber (2005a); 'OWL FA: A Metamodeling Extension of OWL DL'; Proceedings of the International Workshop on OWL: Experience and Directions, OWL-ED2005.

Pan J., G. Stoilos, G. Stamou, V. Tzouvaras and I. Horrocks (2005b); 'f-SWRL: A Fuzzy Extension of SWRL (Extended Version)'; Journal of Data Semantics, Special Issue on Emergent Semantics.

Pan J. and I. Horrocks (2005c); 'OWL-Eu: Adding Customised Datatypes into OWL (Extended Version), Journal of Web Semantics, 4(1).

Pokraev S. and R. Brussee (2005); 'Extending OWL With Maximal Sub-Properties: An Approach to Define Qualified Cardinality Restrictions and Reflexive Properties'; Proceedings of Workshop on OWL: Experiences and Directions (OWL-2005), November 11-12.

Rogers J. and A. Rector (1996); 'The GALEN Ontology'; in Medical Informatics Europe (MIE 96), Copenhagen, IOS Press (pp. 174-178).

Rosse C., I. Shapiro and J. Brinkley (1998); 'The Digital Anatomist Foundational Model: Principles for Defining and Structuring its Concept Domain'; Journal of the American Medical Informatics Association (1998 Fall Symposium Special Issue) (pp. 820-824).

Steels L. (1979); 'Procedural Attachment CSAIL Publications'; AIM-543.

Szolovits P. and L. Ohno-Machado (2005); 'Updating the QMR in 2005: New Approaches'; Harvard-MIT Division of Health Sciences and Technology.

Riazanov A. and A. Voronkov (2002); 'The Design and Implementation of Vampire'; AI Communications, 15(2-3) (pp. 91—110).

Reiter R. (1991); 'The Frame Problem in the Situation Calculus: A Simple Solution (Sometimes) and a Completeness Result for Goal Regression'; in Artificial Intelligence and Mathematical Theory of Computation, Academic Press (pp. 359-380).

Stoilos G., G. Stamou, V. Tzouvaras, J. Pan and I. Horrocks (2005); 'Fuzzy OWL: Uncertainty and the Semantic Web'; Proceedings of the International Workshop on OWL: Experience and Directions, OWL-ED2005.

Tsarkov D., A. Riazanov, S. Bechhofer and I. Horrocks (2004); 'Using Vampire to Reason With OWL'; Proceedings of the 3rd International Semantic Web Conference, ISWC2004, Hiroshima, Japan.

Turing A. (1936); 'On Computable Numbers with an Application to the Entscheidungsproblem'; Proceedings London Mathematical Society, 42 (pp. 230-65).

Waldinger R. (2005); 'Axiomatic Semantics of OWL: Formulation and Validation'; AI Center, SRI International and the Kestrel Institute. http://www.ai.sri.com/daml/owl/axiomatic.htm

Woods W. (1985); 'What's in a Link: the Foundation for Semantic Networks'; in Brachman R. and J. Levesque (eds.) Readings in Knowledge Representation, Morgan Kaufmann.

Wroe C. and J. Cimino (2001); 'Using openGALEN Techniques to Develop the HL7 Drug Formulation Vocabulary'; in American Medical Informatics Association Fall Symposium, AMIA-2001, (pp. 766-770).

Zou Y., T. Finin and H. Chen (2004); 'F-OWL: An Inference Engine for the Semantic Web'; Lecture Notes in Computer Science, Proceedings of the Third International Workshop (FAABS), April 16-18, Greenbelt, Maryland, Springer-Verlag.

Additional Web Sites:

[http://www.w3.org/TR/owl-features/]

[http://www.w3.org/TR/owl-ref]

[http://www.w3.org/TR/owl-semantics/]

[http://www.w3.org/TR/2004/REC-owl-features-20040210/#s1.3]

[http://www.w3.org/TR/2004/REC-owl-features-20040210/#s1.3]

[http://www.w3.org/TR/owl-ref/#OWLDL]

[http://www.w3.org/TR/owl-ref/#Sublanguage-def]

InterSymp-90 Proceedings of the focus symposium on 'Knowledge-Based Systems in Building Design' (ed. J. Pohl); InterSymp-90 5th International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, Aug.7-8, 1990.

InterSymp-91 Proceedings of the focus symposium on 'Computer-User Partnerships in Design' (ed. J. Pohl); InterSymp-91 3rd International Symposium on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, Aug.13-14, 1991.

InterSymp-92 Proceedings of the focus symposium on 'Advances in Computer-Based Design Environments' (ed. J. Pohl); InterSymp-92 6th International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, Aug.18-19, 1992.

InterSymp-93 Proceedings of the focus symposium on 'Advances in Computer-Assisted Building Design Systems' (ed. J. Pohl); InterSymp-93 4th International Symposium on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, Aug.2-5, 1993.

InterSymp-94 Proceedings of the focus symposium on 'Advances in Computer-Based Building Design Systems' (ed. J. Pohl); InterSymp-94 7th International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, Aug.15-21, 1994.

InterSymp-95 Proceedings of the focus symposium on 'Advances in Cooperative Computer-Assisted Environmental Design Systems' (ed. J. Pohl); InterSymp-95 8th International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, Aug. 16-20, 1995.

InterSymp-96 Proceedings of the focus symposium on 'Advances in Cooperative Environmental Design Systems' (ed. J. Pohl); InterSymp-96 International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, Aug.14-18, 1996.

InterSymp-97 Proceedings of the focus symposium on 'Collaborative Design and Decision-Support Systems' (ed. J. Pohl); InterSymp-97 International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, Aug.18-23, 1997.

InterSymp-98 Proceedings of the focus symposium on 'Collaborative Decision- Support Systems for Design, Planning, and Execution' (ed. J. Pohl); InterSymp-98 International Conference on Systems Research, Informatics, and Cybernetics, Baden-Baden, Germany, Aug.17-21, 1998.

InterSymp-99 Proceedings of the focus symposia on 'Computer-Based and Web-Based Collaborative Systems' (eds. J. Pohl and T. Fowler); InterSymp-99 International Conference on Systems Research, Informatics, and Cybernetics, Baden-Baden, Germany, Aug.2-6, 1999.

InterSymp-00 Proceedings of the focus symposia on 'Advances in Computer-Based and Web-Based Collaborative Systems' (eds. J. Pohl and T. Fowler); InterSymp-2000, 12th International Conference on Systems Research, Informatics, and Cybernetics, Baden-Baden, Germany, Jul. 31, Aug. 4, 2000.

InterSymp-01 Proceedings of the focus symposia on 'Advances in Computer-Based and Web-BasedCollaborative Systems' (eds. J. Pohl and T. Fowler); InterSymp-2001, 13th InternationalConference on Systems Research, Informatics, and Cybernetics, Baden-Baden, Germany, Jul. 30-Aug. 4, 2001.

InterSymp-02 Proceedings of the focus symposium on ' Collaborative Decision-Support Systems' (ed. J. Pohl); InterSymp-2002, 14th International Conference on Systems Research, Informatics, and Cybernetics, Baden-Baden, Germany, Jul. 29-Aug. 3, 2002.

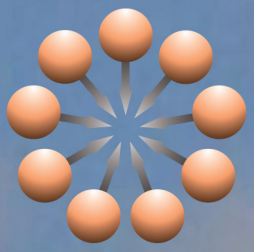
InterSymp-03 Proceedings of the focus symposium on ' Collaborative Decision-Support Systems' (ed. J. Pohl); InterSymp-2003, 15th International Conference on Systems Research, Informatics, and Cybernetics, Baden-Baden, Germany, Jul. 28-Aug. 1, 2003.

InterSymp-04 Proceedings of the focus symposium on 'Intelligent Software Systems for the New Infostructure' (ed. J. Pohl); InterSymp-2004, 16th International Conference on Systems Research, Informatics, and Cybernetics, Baden-Baden, Germany, Jul. 29-Aug. 5, 2004.

InterSymp-06 Proceedings of the focus symposium on 'Advances in Intelligent Software Systems' (ed. J. Pohl); InterSymp-2006, 18th International Conference on Systems Research, Informatics, and Cybernetics, Baden-Baden, Germany, Aug. 7-Aug. 12, 2006.

InterSymp-07 Proceedings of the focus symposium on 'Representation of Context in Software Data -> Information -> Knowledge' (ed. J. Pohl); InterSymp-2007, 19th International Conference on Systems Research, Informatics, and Cybernetics, Baden-Baden, Germany,July 30-Aug. 4, 2007.

InterSymp-08 Proceedings of the focus symposium on 'Intelligent Software Tools and Services" (ed. J. Pohl); InterSymp-2008, 20th International Conference on Systems Research, Informatics, and Cybernetics, Baden-Baden, Germany,July 24-July 30, 2008.



CADRC

Collaborative Agent Design Research Center

Cal Poly
San Luis Obispo, CA 93407 USA

Phone: (805) 756-1310/2841
FAX: (805) 756-7568

e-mail: jpohl@calpoly.edu
www.cadrc.calpoly.edu

ISBN 978-1-897546-00-0