InterSymp – 2007

19th International Conference on Systems Research, Informatics and Cybernetics

(July 30-August 4, 2007, Baden-Baden, Germany)

Pre-Conference Proceedings of the Focus Symposium
on

Representation of Context in Software: Data ➡ Information ➡ Knowledge

Tuesday, July 31, 2007

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: 1-897233-02-7

Preface

For the past 20 years commercial corporations and government agencies have suffered under the limitations of stove-piped computer software applications that function as discrete entities within a fragmented data-processing environment. In the United States of America (US) military services, lack of interoperability has been identified by numerous think tanks, advisory boards, and studies, as the primary information systems problem (e.g., Army Science Board 2000, Air Force SAB 2000 Command and Control Study, and NSB Network-Centric Naval Forces 2000). Yet, despite this level of attention, all attempts to achieve *interoperability* within the current *data-centric* information systems environment have proven to be expensive, unreliable, and generally unsuccessful.

The Quest for *Interoperability*

The expectations of true interoperability are threefold. First, interoperable applications should be able to integrate related functional sequences in a seamless and user transparent manner. Second, this level of integration assumes the sharing of *information* from one application to another, so that the results of a functional sequence are automatically available and similarly interpreted by the other application. And third, any of the applications should be able to enter or exit the integrated interoperable environment without jeopardizing the continued operation of the other applications. These conditions simply cannot be achieved by computer software that processes numbers and meaningless text with predetermined algorithmic solutions through hard-coded dumb data links.

Past approaches to interoperability have basically fallen into three categories. Attempts to create common architectures have largely failed because this approach essentially requires existing systems to be re-implemented in the common (i.e., new) architecture. Attempts to create bridges between applications within a confederation of linked systems have been faced with three major obstacles. First, the large number of bridges required (i.e., the square of the number of applications). Second, the fragility associated with hard-coded inter-system data linkages and, third, the cost of maintaining such linkages in a continuously evolving information systems environment. The third category of approaches has focused on achieving interoperability at the interface boundary. For anything other than limited presentation and visualization capabilities, this approach cannot accommodate dynamic data flows, let alone constant changes at the more useful information level.

These obstacles to interoperability and integration are largely overcome in an *information-centric* software systems environment by embedding in the software some understanding of the information being processed. The term *information-centric* refers to the representation of information in the computer, not to the way it is actually stored in a digital machine. This distinction 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, bound into volumes, folded, and so on. However, all of this is unrelated to the content of the written note on the paper. This content represents the meaning of the sheet of paper. It constitutes the purpose of the paper and governs what we do with the sheet of paper (i.e., its use). In other words, the nature and efficiency of the storage medium is more often than not unrelated to the content or representation that is stored in the medium.

In the same sense, the way in which we store bits (i.e., 0s and 1s) in a digital computer is unrelated to the meaning of what we have stored. When computers first became available they were exploited for their fast, repetitive computational capabilities and their enormous storage capacity. Application software development progressed rapidly in a *data-centric* environment. Content was stored as data that were fed into algorithms to produce solutions to predefined problems in a static problem solving context. It is surprising that such a simplistic and artificially contrived problem-solving environment was found to be acceptable for several decades of intensive computer technology development.

When we established the Collaborative Agent Design Research Center at Cal Poly in 1986, we had a vision. We envisioned that users should be able to sit down at a computer terminal and solve problems collaboratively with the computer. The computer should be able to continuously assist and advise the user during the decision-making process. Moreover, we postulated that one should be able to develop software modules that could spontaneously react in near real-time to changing events in the problem situation, analyze the impact of the events, propose alternative courses of action, and evaluate the merits of such proposals. What we soon discovered, as we naively set out to develop an intelligent decision-support system, is that we could not make much headway by focusing on the representation of data without *context* in a dynamically changing problem environment.

Initially focusing on engineering design, we had no difficulties at all developing a software module that could calculate the daylight available inside a room, as long as we specified to the computer the precise location and dimensions of the window, the geometry of the room, and made some assumptions about external conditions. However, it did not seem possible for the computer to determine on its own that there was a need for a window and where that window might be best located. The ability of the computer to make these determinations was paramount to us. We wanted the computer to be a useful assistant that we could collaborate with as we explored alternative design solutions. In short, we wanted the computer to function *intelligently* in a dynamic environment, continuously looking for opportunities to assist, suggest, evaluate, and, in particular, alert us whenever we pursued solution alternatives that were essentially not practical or even feasible.

We soon realized that to function in this role our software modules had to be able to *reason*. However, to be able to reason the computer needs to have something akin to *understanding* of the *context* within which it is supposed to reason. The human cognitive system builds context from knowledge and experience using *information* (i.e., data with attributes and relationships) as its basic building block. Interestingly enough the storage medium of the information, knowledge and context held by the human brain is billions of neurons and trillions of connections (i.e., synapses) among these neurons that are as unrelated to each other as a pencilled note and the sheet of paper on which it is stored.

What gives meaning to the written note is its *representation* within the framework of a language (e.g., English) that can be understood by the reader. Similarly, in a computer we can establish the notion of *meaning* if the stored data are represented in an ontological framework of objects, their characteristics, and their interrelationships. How these objects, characteristics and relationships are actually stored at the lowest level of bits (i.e., 0s and 1s) in the computer is immaterial to the ability of the computer to undertake reasoning tasks. The conversion of these bits into data and the transformation of data into information, knowledge and context takes place at higher levels, and is ultimately made possible by the skillful construction of a network of richly described objects and their relationships that represent those physical and conceptual aspects of the real world that the computer is required to reason about.

This is what is meant by an information-centric computer-based decision-support environment. One can further argue that to refer to the ability of computers to *understand* and *reason* about *information* is no more or less of a trick of our imagination than to refer to the ability of human beings to understand and reason about information. In other words, the countless minuscule charges that are stored in the neurons of the human nervous system are no closer to the representation of information than the bits (i.e., 0s and 1s) that are stored in a digital computer. However, whereas the human cognitive system automatically converts this collection of charges into information and knowledge, in the computer we have to construct the framework and mechanism for this conversion. Such a framework of objects, attributes and relationships provides a system of integrated software applications with a common language that allows software modules (now popularly referred to as *agents*) to *reason* about events, monitor changes in the problem situation, and collaborate with each other as they actively assist the user(s) during the decision-making process. One can say that this *ontological framework* is a virtual representation of the real world problem domain, and that the agents are dynamic tools capable of pursuing objectives, extracting and applying knowledge, communicating, and collaboratively assisting the user(s) in the solution of current and future real world problems.

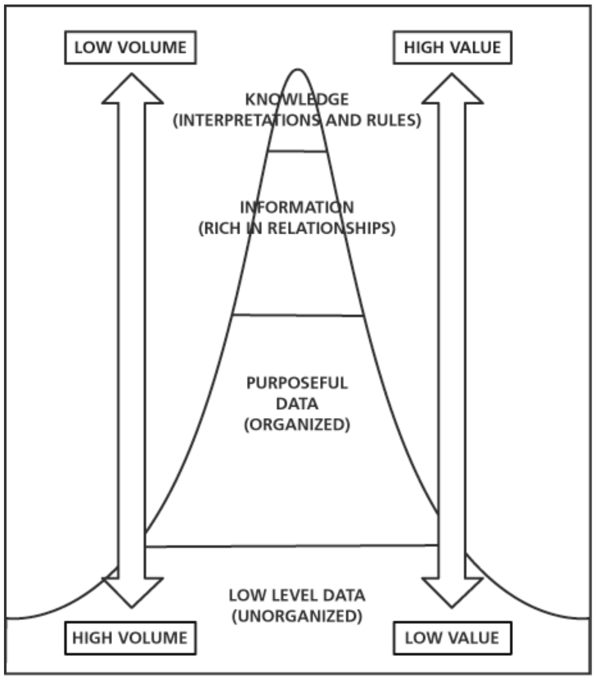


Figure A: The information overload myth

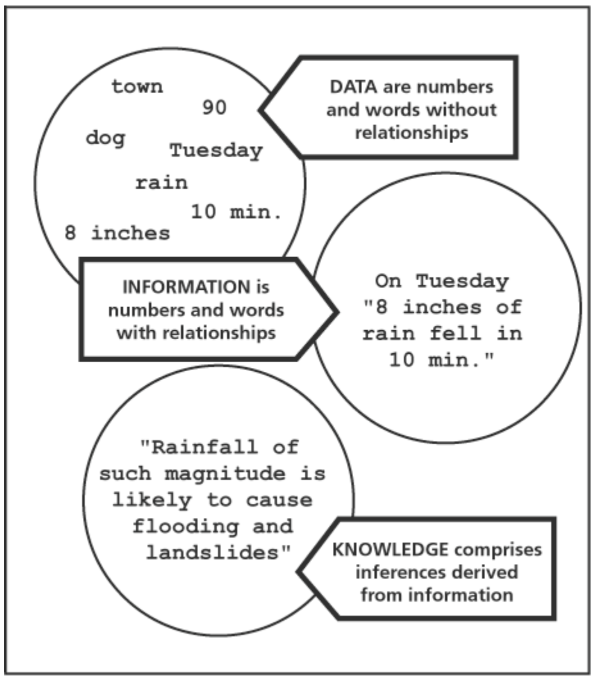

town
90
dog
Tuesday
rain
10 min.
8 inches
DATA are numbers and words without relationships
INFORMATION is numbers and words with relationships
On Tuesday "8 inches of rain fell in 10 min."
"Rainfall of such magnitude is likely to cause flooding and landslides"
KNOWLEDGE comprises inferences derived from information

Figure B: Data, information and knowledge

Data, Information, and Knowledge

It is often lamented that we human beings are suffering from an *information overload*. This essentially a myth, as shown in Figure A. Instead we are suffering from a data overload. The confusion between data and information is not readily apparent and requires further explanation.

Unorganized data are voluminous but of very little value. Over the past 20 years, industry and commerce have made significant efforts to rearrange this unorganized data into purposeful data, utilizing various kinds of database management systems. However, even in this organized form, we are still dealing with data and not information.

Data are defined as numbers and words without relationships. In reference to Figure B, the words *town*, *dog*, *Tuesday*, *rain*, *inches*, and *min*, have little if any meaning without relationships. However, linked together in the sentence: "On Tuesday, 8 inches of rain fell in 10 min.", they become information. If we then add the context of a particular geographical region, pertinent historical climatic records, and some specific hydrological information relating to soil conditions and behavior, we could perhaps infer that: "*Rainfall of such magnitude is likely to cause flooding and landslides.*" This becomes knowledge.

Jens Pohl, May 2007

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

InterSymp-2007
International Conference on Systems Research, Informatics and Cybernetics

Focus Symposium
on
Representation of Context in Software:
Data ➡ Information ➡ Knowledge
Tuesday, July 31, 2007

TABLE OF CONTENTS

Knowledge Management Enterprise Services (KMES)

Concepts and Implementation Principles

Jens Pohl, Ph.D.

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

Abstract

The purpose of this paper is to present concepts and implementation principles related to the design and development of reusable software services that are capable of assisting users at the operational level. Knowledge Management Enterprise Services (KMES) are an implementation of the service-oriented architecture paradigm, with a focus on the exchange of data within the meaningful context of a particular application (i.e., knowledge) domain. This requires a KMES service to incorporate a high level representation of this knowledge domain in the form of an ontology that is shared among all collaborating services within the application environment and at the same time specialized to the perspective that is appropriate to the servicing capabilities of the particular KMES service.

Although KMES services can operate in any distributed system environment, they represent a step toward semantic web services by incorporating many of the same objectives, such as self-sufficiency, interoperability, discovery, asynchronous interaction with clients, and context-based intelligence. Therefore, this paper also deals briefly in an Appendix with the notion of *web enabled* and the different types of thin-client user-interfaces that are prevalent today.

Finally, the paper discusses the software development process of a software system environment that maximizes the use of KMES services. Based on our CADRC Center's experience with the development of mostly military decision-support systems incorporating collaborative software agents, such KMES-based systems offer several advantages including a significant reduction in development time, decreased software development costs, and higher quality end-products.

Keywords

agents, context, data, data-centric, decision-support, design, development process, evaluation, information, information-centric, intelligence, KMES, knowledge management enterprise services, ontology, representation, semantic web, service-oriented architecture, software, thin-client, web enabled, web services.

The Service-Oriented Architecture Paradigm

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. In this way the adapter that is required when you take your laptop computer on a business trip to another country interfaces between the foreign electrical outlet and the electric plug of your computer, which in turn interfaces with the electric cables that interface with the power unit, and so on.

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 came along that these concepts could be readily implemented (Erl 2005). Initial attempts to provide the required communication infrastructure, such as the Distributed Computing Environment (DCE) and the Common Object Request Broker Architecture (CORBA) did not gain the necessary general acceptance (Mowbray and Zahavi 1995, Rosenberry et al. 1992). Web services and SOA are similar in that they both support the notion of discovery (Gollery 2002). Web services employ the Universal Description Discovery and Integration (UDDI) mechanism for providing access to a directory of web services, while SOA services are published in the form of an Extensible Markup Language (XML) interface.

So what is SOA? 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)[1] 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 Existing Web Services Environment

Web services are a particular implementation of the SOA paradigm. According to the World Wide Web Consortium (W3C) a web service may be defined as "… *a software application identified by a Uniform Resource Identifier (URI), whose interfaces and bindings are capable of*

[1] 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.

being defined, described, and discovered by XML artifacts"[2]. Currently most web services interact with other services or users utilizing the Hyper-Text Transfer Protocol (HTTP) to exchange XML-based messages defined in the Service Oriented Architecture Protocol (SOAP). The SOAP standard defines an XML language and a set of rules for formatting objects and data that are independent of the programming language, operating system, and hardware platform.

Existing web service environments such as Microsoft's '.Net' software (Thai 2003, Chappell 2006) typically comprise a web server that utilizes HTTP for communication, UDDI as part of the standard definition of web service registries, a registry that already contains an entry for the accessing application, and any number of web services designed to facilitate some of the operations that the accessing application may wish to perform. In this respect, current web service environments rely on the notion of *predefined registrations and discovery* and do not support the notion of *opportunistic discovery*. UDDI, an international standard that defines a set of methods for accessing a registry, is structured to provide information about organizations, such as: who (about the particular organization); what (what services are available); and, where (where are these services available). However, UDDI does not provide descriptions of the available services in a semantic form that can be automatically interpreted by software (e.g., software agents), rather the descriptions are hard-coded or subject to human interpretation.

Communication between an application and a web server is almost always initiated by the application (i.e., the application sends a request and the web server sends a response). Specifically, the Uniform Resource Locator (URL) contains an identification of the particular web server to be used. This web server then finds the HTML page that corresponds to the URL and returns that page in the response. Immediately after the response has been sent the connection between the application and the web server is terminated and only reactivated if another response is requested. In this way a web server is able to handle many concurrent requests from applications.

Adding Meaning to Web Services

There are several reasons why computer software, and therefore web services, 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, rather than the *context* within which the collected data would later become useful in planning, monitoring, assessment, and decision-making tasks.

What are the enabling facilities that will allow software to interpret the meaning of data as an intelligent partner to the human user? This is a question that has engaged our CADRC Center in intensive explorations for the past 20 years. Several years before the advent of the Internet and the widespread promulgation of SOA concepts we started building distributed software systems

[2] See web site at: http://www.w3.org/TR/wsa-reqs#IDAIO2IB.

of loosely coupled modules that were able to collaborate by subscription to a shared information model. Today, our KMES components are based on the same foundational principles to enable them to function as decoupled services. These principles include:

- An internal *information* model that provides a usable representation of one or more of the following three information areas: (a) the application domain in which the service is being offered; (b) the internal operational domain of the software application itself; and, (c) the role of the service within the external environment. In other words, the context provided by the internal information model must be adequate for the software application to perform as a useful adaptive set of tools in its area of expertise, be able to monitor and diagnose its own internal operational state, and describe its nature in response to external inquiries.
- The ability to *reason* about events within the context provided by the internal information model. Eventually these reasoning capabilities should extend beyond the ability to render application domain related services and perform self-monitoring maintenance and related operational efficiency tasks, to the ability of a service to be able to describe its capabilities and understandings to other external parties (i.e., other services and human users).
- Facilities that allow the service to *discover* other services and understand the nature and capabilities of these external resources.
- 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. In other words, the internal information model must be dynamically *extensible*.
- 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. A typical relatively simple intent might be the objective of finding another service application capable of providing a specific service such as a weather forecast or an available weapon for destroying a given target. Far more sophisticated would be an objective that is only vaguely defined, such as the solution of a problem for which the solution approach is known in general terms only (e.g., locate the likely position of an enemy unit that has not been sighted for 24 hours).
- The ability of a service to increase its capabilities by either *generating* new tools (e.g., creating new agents or cloning existing agents) or *searching* for external assistance.

The Concept of Knowledge Management Enterprise Services

Knowledge Management Enterprise Services (KMES) are *self-contained* software components that offer their capabilities as services to external service requestors. Whereas in a SOA-based software system the available services normally operate at a lower system level as enablers of higher level functional capabilities, a KMES component is the incarnation of one or more of those functional capabilities. In other words, KMES components are services that operate at the functional level of the application domain.

They are designed to be platform independent and adaptable to a variety of applications. It is this adaptability that promotes their high degree of *reusability*. Some KMES components may have quite narrow capabilities such as the reformatting of weather data into a software interpretable weather forecast, while others will incorporate larger functional domains such as the optimum routing of goods from multiple origins along alternative routes to multiple destinations. However, all of the services that operate within a given application domain are closely aligned to the knowledge context of that domain by sharing the same information model. While the software code of a KMES component is reusable, its internal information model needs to be reconfigured as it is moved from one application domain to another. This ensures that the KMES services within any particular application environment are able to exchange data within the functional context of that environment.

For example, in the transportation domain the optimum routing of goods from multiple origins along alternative routes to multiple destinations would include the following KMES components, providing their services within the common context of a shared information model:

- Conveyance load-planning (i.e., ships, barges, trucks, railcars, and aircraft of various types).
- Packaging of different kinds of shipping units (e.g., containers, pallets).
- Storage management in marshalling yards and warehouses.
- Route planning and re-planning.
- Map-based presentation for geospatial tracking.
- Scheduling.
- Interoperability bridges to external data feeds and other applications.
- Graphical and textual report generators.

This is in stark contrast to the large software systems that have been developed in the past and that invariably lead to a stove-piped architecture with almost insurmountable interoperability problems. Typically, in the case of these legacy systems the above functional capabilities have required the development of several systems with considerable duplication (e.g., user-interfaces, persistence facilities, and report generation) and largely incompatible data schemas.

A KMES-based system governed by SOA principles, on the other hand, is intended to meet several technical objectives that are aimed at maximizing horizontal and vertical interoperability. First and foremost, a KMES is designed to be as self-sufficient as the state of current technology will allow. Ideally, self-sufficiency should include platform independence with self-installing, self-configuring, and self-scaling capabilities. Second, it must incorporate discovery capabilities. However, discovery capabilities that are truly useful will require some degree of built-in intelligence. The combination of self-sufficiency, discovery and intelligence is potentially very powerful since it supports interoperability at the information level. In other words, KMES components are able to exchange data in the *context* that is provided by the shared virtual model of a particular real world knowledge domain.

Third, a KMES incorporates intelligent tools in the form of agents that support meaningful human-to-agent and agent-to-agent collaboration. These agents rely on the context provided by the internal information model to an even greater degree than do the discovery capabilities. In both cases the availability of context is a prerequisite for automated reasoning capabilities that can be applied to the interpretation of data changes, the opportunistic analysis of events, the spontaneous search for additional resources, and the generation of warnings and alerts.

Fourth, a KMES is capable of exposing functionality through objectified, domain-centric client interfaces. To take full advantage of this capability the KMES must support asynchronous interaction with its external clients. This in turn requires adherence to industry-standard patterns such as JavaBeans[3], Property Change Management[4], and so on.

Fifth, by virtue of its internal information model a KMES is readily adaptable to operate in terms of application-specific notions and concerns. However, this also means that when reference is made to the *reusability* of a KMES then this refers to the software code and not the internal information model. Reconfiguring a KMES to a new application domain will involve initialization with the ontology of the new knowledge domain. Capitalizing on their decoupled nature, KMES components can be replaced with improved versions as the technology advances.

In summary, KMES modules are adaptable, self-contained software components that are capable of performing specific tasks within a net-centric environment. Service-oriented KMES capabilities typically take the form of distributable services whose functionality is exposed to potential clients as domain-centric objects employing key industry-standards. In other words, interaction between KMES services and their clientele occurs in terms of object-level operations (i.e., object creation, property modification, etc.) over the domain model exposed by such services. Such object-level manipulation is partnered with the asynchronous notification of events to interested parties (i.e., clients as well as the KMES service itself). For example, consider a KMES Route Planning service. Such a service could be invoked by a client creating a set of domain objects (e.g., *requirements*, *constraints*, and *route topology* objects) defining the context of their request. Listening to such objects, the KMES Route Planning service responds by processing this context into a solution. This solution would, in turn, be exposed as a set of objects based on the service's domain model interface. In the same asynchronous manner by which the service became aware of the client's initial request, the client in turn will receive its results through object-level event notification.

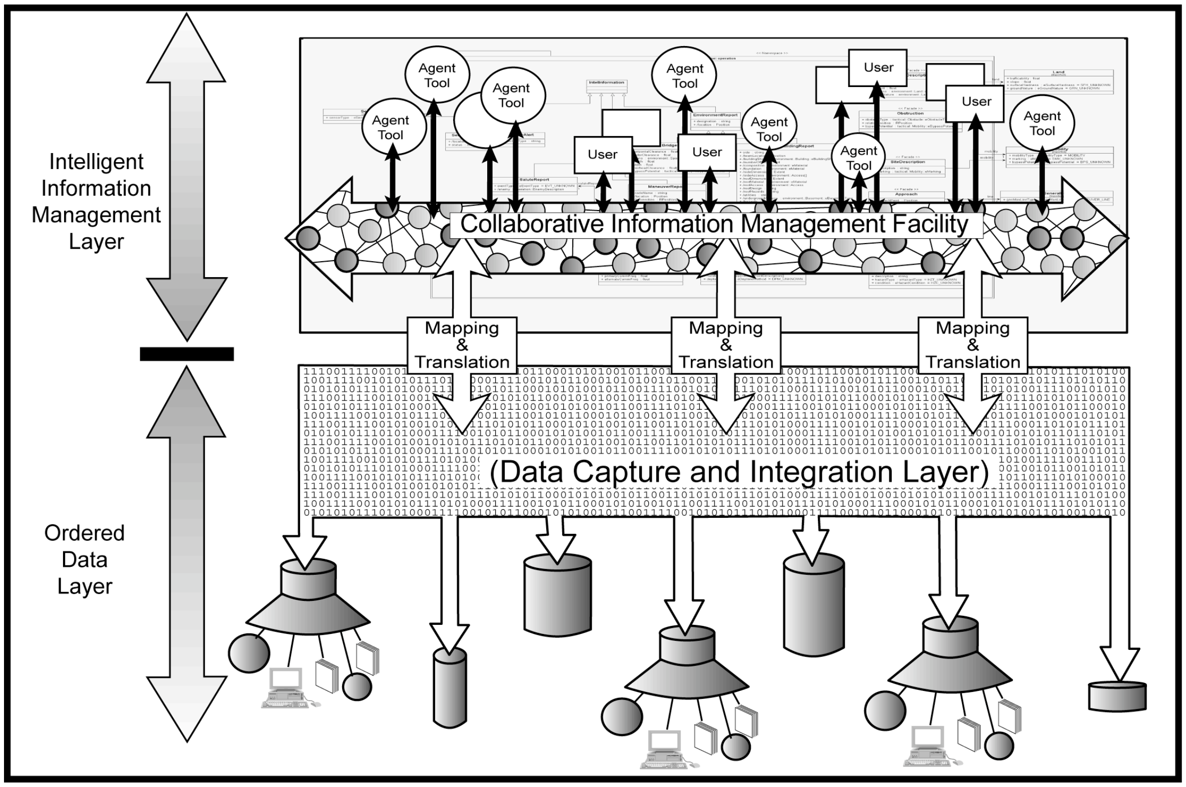
KMES as a Net-Centric Architecture

The expressive, context-rich representation upon which many KMES capabilities are built together with the significant potential for higher levels of decision-support lends itself to incorporation of intelligent agent technology. When equipped with such enabling features, agents can collaborate with users to assist in formulating solution alternatives, compare and contract their associated costs, and aid in successful execution through constant monitoring and the performance of necessary mediation. For example, agents in a military logistical domain can receive status reports, track shipments, incorporate suitable and available assets in plans, and provide appropriate updates on location and security risks. Others may track the path of incidence and provide appropriate graphic and textual updates for action. Finally, agents can interpret incoming signals, identify significant events (i.e., changes), and modify proposals to

[3] JavaBeans are reusable software components (i.e., classes) written in the Java computer language. Based on certain conventions (i.e., naming, construction and behavior) a JavaBean is used to encapsulate several objects into a single object that can then be passed around.

[4] Property Change Management refers to the ability to access diverse documents across a network (i.e., Internet or intranet) without the need for manual intervention. For example, the Web Interface Definition Language (WIDL) automates interactions with HTML/XML documents and thereby allows the Web to serve as an integration platform.

meet the changing situation as it develops. The vision of such *intelligent agents* is quite compelling and it is now generally believed to be a critical component for successfully harnessing the increasing complexities of a *net-centric* environment.



Intelligent Information Management Layer
Ordered Data Layer
Agent Tool
User
Collaborative Information Management Facility
Mapping & Translation
(Data Capture and Integration Layer)

Figure 1: Conceptual KMES-based net-centric architecture

Existing data-centric systems lacking the adaptive, interoperability characteristics described above can be integrated into such an agent-empowered KMES software environment through the use of *interoperability bridge facilities* that effectively map the data model in one system to the information model of the other. This allows for a meaningful bi-directional interaction and exchange of data in context. Such bridges have been successfully demonstrated by military organizations for linking legacy data-centric systems to intelligent command and control systems (Pohl et al. 2001). The technology is inherently scalable and allows for the efficient and effective interconnection of multiple participants within a heterogeneous net-centric environment.

Conceptually, an intelligent net-centric software environment typically requires the seamless integration of a KMES-based information management facility with existing data sources. This can be achieved with an *information-centric* architecture that consists essentially of two components (Figure 1): a data-centric Data Capture and Integration Layer that incorporates linkages to existing data sources; and, an Intelligent Information Management Layer that overlays the data layer and utilizes software agents with automatic reasoning capabilities, serving as decision-support tools.

The Intelligent Information Management Layer architecture (Figure 1) utilizes intelligent software agents capable of collaborating with each other and human operators in planning, re-planning, monitoring, and associated decision-support environments. Typically such intelligent systems are based on software development frameworks, such as the ICDM (Integrated Cooperative Decision Making) and TIRAC™ (Toolkit for Information Representation and Agent Collaboration) software development frameworks used by CDM Technologies and the Collaborative Design Research Center (CADRC) at Cal Poly[5] for the development of military and commercial systems, respectively (Pohl et al. 2004a and 2004b).

Data Capture and Integration Layer: The bottom layer of the system takes the form of an operational data store and/or Data Warehouse, implemented within a commercial off-the-shelf relational database management system (RDBMS). This repository integrates data extracted on a periodic basis from several external sources into a *common* data schema. Although not a requirement, the design of the data schema is typically closely modeled on the structure of the ontology of the Intelligent Information Management Layer to minimize the required data-to-information and information-to-data mappings between these two system layers. Further, to facilitate an object-oriented environment, content managed by the Data Capture and Integration Layer is exposed to its information-oriented clients (e.g., KMES-based environments) as objects rather than relational tables. Translation between these two forms is typically accomplished through employment of some form of Object Relational Mapping (ORM) technology.

In conformity with normal enterprise data management practices the Data Capture and Integration Layer incorporates the following four characteristics:

- It is subject-oriented to the specific business processes and data domains relevant to the application area (e.g., goods movement across national borders or tactical command and control in a military theater).
- It is integrated so that it can relate data from multiple domains as it serves the data needs of the analysis functions performed by collaborative agents in the Intelligent Information Management Layer.
- It is periodically synchronized with events and changes occurring in the external data sources from which it derives its content.
- It is time-based to support the performance of analyses over time, for the discovery of patterns and trends.

A multi-tier architecture is used to logically separate the necessary components of the data layer into levels. The first tier is the RDBMS, which ensures the persistence of the data level and provides the necessary search, persistence, and transaction management capabilities. The second tier is the service level, which provides the interface to the objectified data level and at the same time supports the data access requests that pass through the mapping interface from the Intelligent Information Management Layer to the Data Capture and Integration Layer. It is designed to support request, response, subscribe, and publish functionality. The third tier is the control level, which routes information layer and user requests to the service level for the update, storage and retrieval of data. Finally, a view layer representing the fourth tier serves as a graphical user-interface for the Data Capture and Integration Layer.

Information Management Layer: The Intelligent Information Management Layer consists of KMES components in the form of a group of loosely coupled and seamlessly integrated

[5] ICDM and TIRAC™ are software development toolkits developed by principals of the CADRC Center at California Polytechnic State University, San Luis Obispo and its commercial arm CDM Technologies, Inc..

decision-support tools. The core element of each KMES component is typically an ontology that provides a relationship-rich and expressive model of the particular domain over which the KMES capability operates. Normally, KMES components are based on a three-tiered architecture incorporating technologies, such as distributed-object servers and inference engines, to provide a framework for collaborative, agent-based decision-support that offers developmental efficiency and architectural extensibility. The multi-tiered architecture clearly distinguishes between information, logic, and presentation. Most commonly an *information tier* consists of a collection of information management servers (i.e., information server, subscription server, etc.) providing domain-oriented access to objectified context, while a *logic tier* houses communities of intelligent agents, and a *presentation tier* is responsible for providing meaningful interfaces to human operators and external systems.

A Typical KMES Ontology

This section discusses a portion of a domain-centric ontology upon which a particular logistics-oriented KMES capability may operate. It should be made clear, however, that while in this case the ontology deals with logistic operations, the domain, scope and expressiveness within that domain, as well as the bias (i.e., perspective) of the model is driven by the use-cases that are supported by the KMES capability, as well as the subject matter those use-cases operate over. In the example provided below, the underlying KMES ontology is divided into several somewhat related domains (Figure 2). While some of these domains describe application-specific events and information (e.g., goods movement transactions, shipping routes, and so on) others describe more general, abstract notions (e.g., event, threat, view, privacy). The goal in developing such an ontology is to abstract general, cross-domain notions into high-level, extensible domain models. As such, these descriptions can be refined and specialized across several application sub-domains. In other words, more domain-specific, concrete notions can be described as extensions of these abstract models.

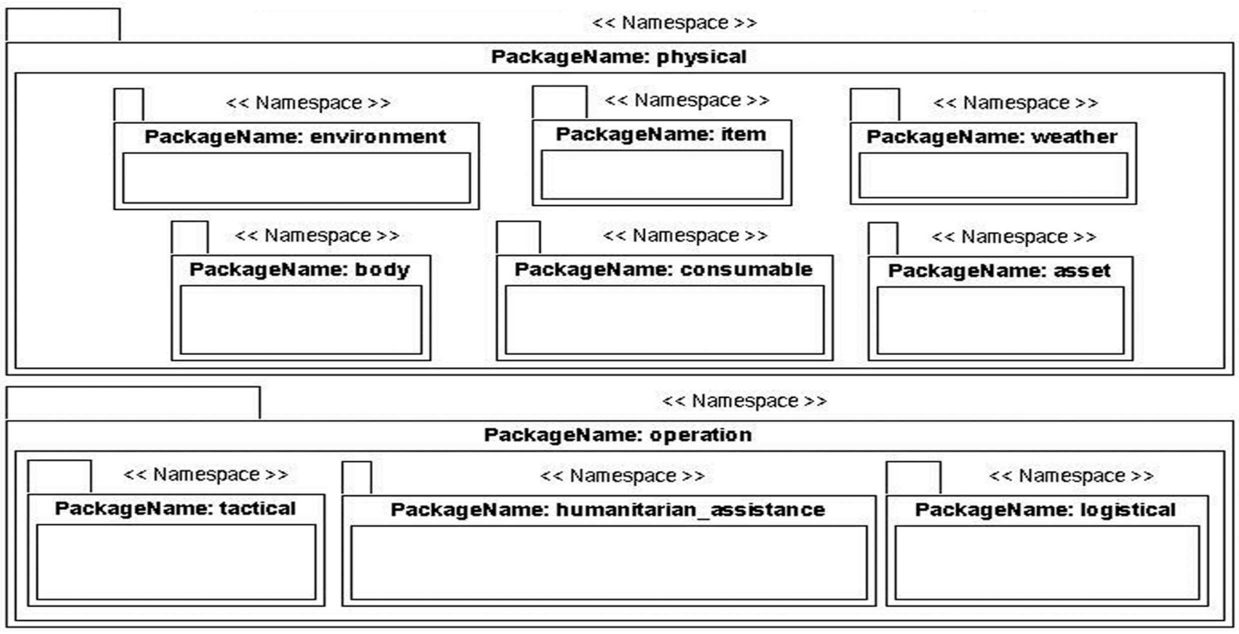


Figure 2: Typical ontology domains within a military application area.

Accordingly, a KMES ontology includes several primary meta-characteristics. Through mechanisms such as inheritance as well as the application of underpinning analysis patterns, these meta-characteristics can be propagated to more specific ontological components. To illustrate, the simple application of inheritance allows, for example the abstract characteristic of something being trackable to be propagated into more specialized entities. Applied to this logistics example, if such a trackability notion is introduced at the *physical.Mobile* level then, through inheritance, any entity that is a kind of *physical.Mobile* automatically receives the property of being *trackable*. Taking this example further, a second meta-characteristic may relate to the dispensability of an item. If this property is represented at the *physical.Item* level then, similar to the *trackable* characteristic, anything that is a kind of a *physical.Item* automatically receives the quality of being dispersible or *suppliable*. In addition, as an extension of *physical.Mobile*, such *suppliable* items are also *trackable*. It can be readily seen that together these two meta-characteristics provide an effective foundation for propagating fundamental notions to hierarchically related definitions. Although inheritance can be a useful mechanism for the propagation of fundamental characteristics to more specific classifications, an even more powerful, and oftentimes less restrictive, technique is the application of extensible analysis patterns. Such patterns offer adaptable model fragments, thereby providing a fundamental definition of the notion being represented in the form of an extensible model architecture for applying this underpinning concept to other elements of a domain model. Such patterns typically employ a *role* metaphor, where elements of a model may essentially *play the role* of something embodying the fundamental notion or characteristic.

The Capabilities of KMES Agents

KMES components equipped with intelligent agents may employ a variety of framework technologies and reasoning paradigms to execute their agent-based logic. Regardless of the specific agent technology employed, their capabilities can exist at a monitoring, largely reactive level, or at a higher consequential and proactive level. In actuality, the event-oriented nature of the former may, in fact, trigger the proactive reasoning of the latter. In the context of homeland security, for example, such reasoning may produce: a warning[6] that hazardous material is en route; a warning that a truck has not reached a waypoint within a certain time limit; an alert that a truck has not reached a waypoint within a more critical time limit; a warning that a truck is near a higher risk area; an alert that a truck has stopped for more than a certain time near a higher risk area; an alert that the loaded weight of a truck does not match the final weight at the border check point; and so on.

Within the same homeland security context (i.e., specifically inland border control), typical higher level agent inferencing capabilities may include warnings and alerts that a particular combination of circumstances involving encyclopedic data and truck-based or convoy-based confirmation data entered at waypoints and checkpoints constitutes a higher risk situation. Examples include, a particular driver transporting certain kinds of goods, or the combination of an authorized substitute driver taking an authorized alternative route without apparent reason, and taking a significantly longer time between two consecutive checkpoints. While none of these

[6] Typically, agents will communicate with the user at different levels of urgency. For example, a warning may simply draw the user's attention to some particular event or situation, while an alert signifies that the user's focused attention is urgently required.

individual anomalies might be sufficient to cause concern, their combined occurrence may well constitute a risk requiring further actions.

Comparing the Development of Legacy and KMES-Based Systems

Considerable time and cost savings can be realized in the KMES approach, without sacrificing quality. In fact, the quality of the software developed can increase due to both the extensive formal validation and verification process appropriate for a core capability as well as the informal validation and verification resulting from its repeated use in the field. This is readily seen when we compare the development life-cycle of a legacy software system (Figure 3) with a KMES-based system (Figure 4).

Software development projects, whether legacy or KMES-based, commence with the recognition of a need. Typically this is a functional need that has been identified by an operational failure or through some form of analysis driven by a desire to achieve a higher level of effectiveness in supporting certain operations. This is followed by the formulation of an end-state vision and, if this vision in conjunction with the need are sufficiently compelling, a decision to act. Once that decision has been made the translation of the end-state vision into a set of use-cases on the basis of which the actual product requirements are formulated. While both the initial level of detail contained in the use-cases, consequential requirements specification, and the degree of involvement of the development team in the formulation of these two artifacts will vary with the type of project and the kind of development process adopted (e.g., prescriptive, agile), there is an undisputed need for some form of formalized documentation describing these various aspects.

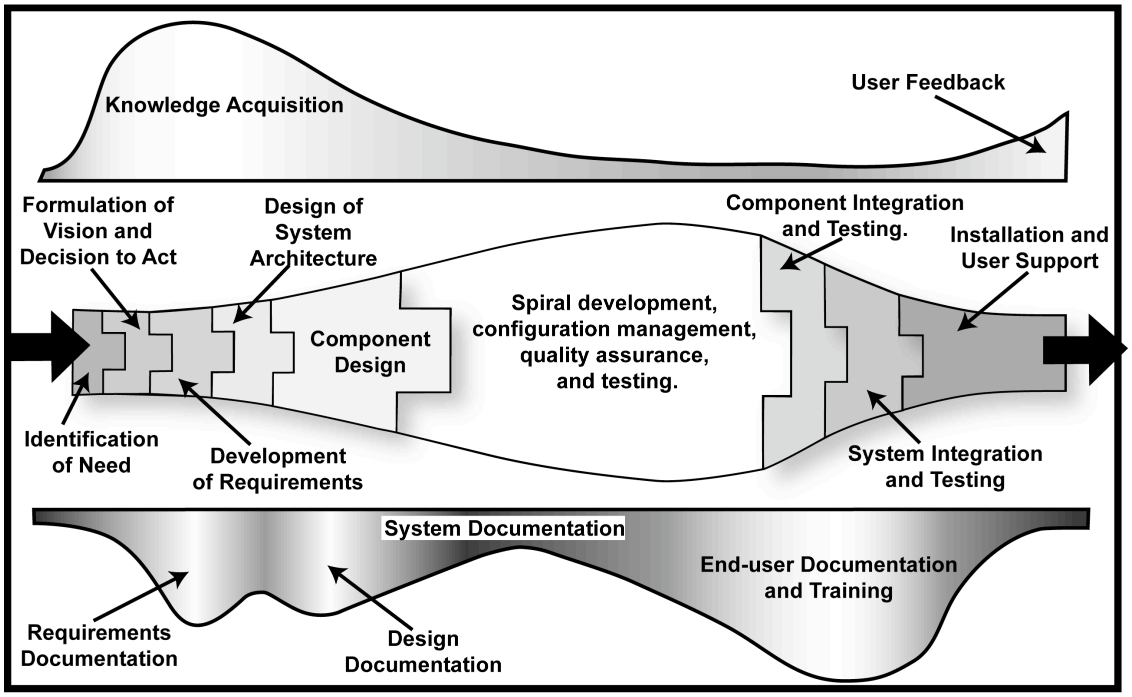

Knowledge Acquisition
User Feedback
Formulation of Vision and Decision to Act
Design of System Architecture
Component Integration and Testing.
Installation and User Support
Spiral development, configuration management, quality assurance, and testing.
Component Design
Identification of Need
Development of Requirements
System Integration and Testing
System Documentation
End-user Documentation and Training
Requirements Documentation
Design Documentation

Figure 3: Development life-cycle of legacy software

Up to this point, as shown in Figures 3 and 4, there appears to be little difference between the legacy and the KMES-based software development approaches. In either case the degree to which the software marketing and development teams will be permitted to assist the customer and end-user in establishing end-state objectives and requirements depends largely on factors that are only indirectly related to the development approach. One of these factors is that as a result of rapid advances in information technology the potential users of this technology are often not aware of the kind of progressive capabilities that could significantly improve the efficiency and effectiveness of their efforts.

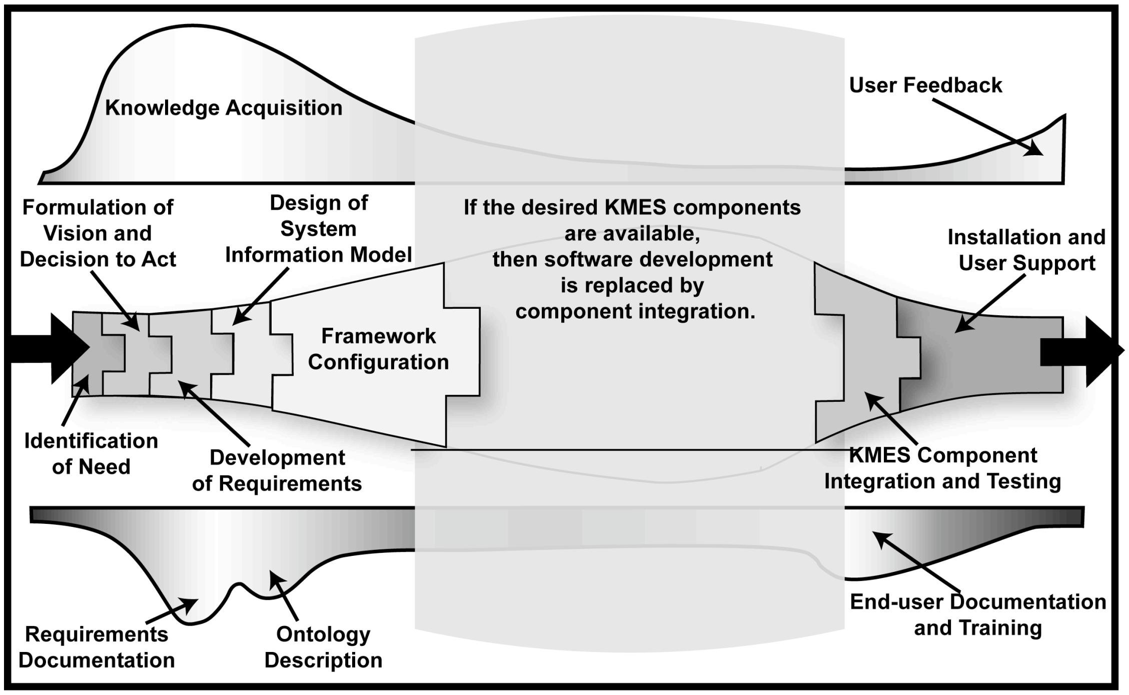

Knowledge Acquisition
User Feedback
Formulation of Vision and Decision to Act
Design of System Information Model
If the desired KMES components are available, then software development is replaced by component integration.
Installation and User Support
Framework Configuration
Identification of Need
Development of Requirements
KMES Component Integration and Testing
End-user Documentation and Training
Requirements Documentation
Ontology Description

Figure 4: Development life-cycle of KMES-based software

Once the requirements have been established the development process of legacy software will normally proceed with the design of the system architecture (Figure 3). However, in the case of a KMES-based solution the mature design and engineering principles have already produced an open architecture exploiting the semantic-rich representation (i.e., *context*) and well-structured interface protocols that will allow the KMES components to effectively interact (Figure 4). Therefore, the substitute step in the KMES-based approach is the design of aspects specific to the application logic, representation, and presentation, each of which may capitalize on supportive building blocks offered by the KMES capabilities being employed. In fact, much of this effort will essentially take the form of *adapting* off-the-shelf KMES capabilities to operate in terms of application-specifics.

In the case of legacy software formulation of application-specific logic, representation, and presentation is in addition to the formulation of the core functionality that would otherwise have been taken care of by inclusion of KMES components. Further, the legacy design and implementation of such core functionality is typically tailored to the specific application leaving little opportunity for reuse. As a result, if the required KMES components are available then the normal time consuming and costly development cycle of traditional software is avoided and replaced by the relatively simple process of integrating KMES components into the target operating environment. Even with the adoption of a spiral development cycle, the traditional software development cycle can take years and account for as much as 60% of the total software production cost and time. In the case of KMES-based software the component integration stage can be completed in a matter of a few months and sometimes weeks. As a result, the delivery of the first set of usable capabilities can be reduced to six months or less after the initiation of the software development project.

KMES Benefits to the Customer

The principal benefits of KMES-based software systems are threefold: early delivery of usable decision-support tools; decreased software acquisition costs; and, higher quality products. In the experience of our Center, which is predominantly engaged in the design and development of intelligent software systems for national security applications, the considerable time savings that can be achieved with the KMES approach has been of particular interest to our military customers. This is probably due to their increasing focus on *adaptive planning* capabilities. In this military context *adaptive planning* is defined by the Adaptive Planning Roadmap[7] as the capability to create and revise plans rapidly and systematically, as circumstances require.

Our military customers quickly realized that the adaptive planning mandate will require new planning and decision-support tools with superior capabilities (i.e., intelligence) that can be rapidly implemented, and are extensible and replaceable to accommodate the evolving needs of the user community. KMES-based software systems have the potential for meeting this challenge by virtue of the following inherent advantages:

- *Rapid delivery* of meaningful capabilities, with the potential of achieving a first usable product installation within three to six months after the initiation of a software development project.
- *Lower cost* due to the replacement of the normally prolonged software development period with a much shorter KMES integration period.
- *Greater reliability* and quality due to exhaustive core-component verification and validation in conjunction with the maturity that comes from extensive in-field use.
- *Interoperability* through component design based on standard protocols and a decoupled, multi-tiered framework.
- *Flexibility* to extend functional capabilities through plug-and-play components and an open architecture.
- *Multiple deployment options* including net-centric delivery alternatives of hosted-managed services.

[7] Adaptive Planning Implementation Team (ODASD/JOWPD); 'Adaptive Planning Roadmap'; Version 1, Final Draft, 3 January 2005 and 'Adaptive Planning and Execution Roadmap III', Draft 8 February 2007, Joint Chiefs of Staff, US Department of Defense.

References

Chappell D. (2006); 'Understanding .NET'; 2nd Edition, Independent Technology Guides, Addison-Wesley, Boston, Massachusetts.

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

Gollery S. (2002); 'The Role of Discovery in Context-Building Decision-Support Systems'; Office of Naval Research Workshop on Collaborative Decision-Support Systems, Quantico, Virginia, 18-19 September (Proceedings available from CADRC Center, Cal Poly, One Grand Avenue (Bdg. 117T), San Luis Obispo, California 93407).

Mowbray T. and R. Zahavi (1995); 'The Essential CORBA: Systems Integration Using Distributed Objects'; Wiley, New York, New York.

Pohl J., M. Porczak, K.J. Pohl, R. Leighton, H. Assal, A. Davis, L. Vempati and A. Wood, and T. McVittie, and K. Houshmand (2001); 'IMMACCS: A Multi-Agent Decision-Support System'; Technical Report, CADRU-14-01, Collaborative Agent Design (CAD) Research Center, Cal Poly, San Luis Obispo, CA, June. (2nd Edition)

Pohl J., K. Pohl, R. Leighton, M. Zang, S. Gollery and M. Porczak (2004a); 'The ICDM Development Toolkit: Purpose and Overview'; Technical Report CDM-16-04, CDM Technologies, Inc., San Luis Obispo, California, USA (May).

Pohl J., K. Pohl, R. Leighton, M. Zang, S. Gollery and M. Porczak (2004b); 'The TIRAC™ Development Toolkit: Purpose and Overview'; Technical Report CDM-17-04, CDM Technologies, Inc., San Luis Obispo, California, USA (August).

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.

Rosenberry W., D. Kenney and G Fisher (1992); 'Understanding DCE'; O'Reilly and Associates, Sebastopol, California.

Thai T. L. (2003); '.NET Framework Essentials'; 3rd Edition, O'Reilly, Sebastopol, California.

Appendix: The Multiple Meanings of *Web Enabled*

The term *web enabled* is widely used with somewhat differing intended meanings. The following explanation of this term was prepared as an internal communication by Steve Gollery, a Senior Software Engineer in our CADRC Center. It is reproduced in abbreviated form in this paper as an appendix for clarification purposes.

> The phrase *web enabled* is sufficiently vague that it provides little guidance in understanding the intent of the persons and organizations using it. This lack of definition becomes critical in view of the fact that some meanings of *web enabled* severely limit the

capabilities of client-side software. The following list of the possible meanings of *web enabled* may not be inclusive, but it does cover the main variations.

1. A *rich client*, communicating with its server(s) using HTTP, SOAP, and so on. In this configuration the capabilities of the rich client are essentially unconstrained since it is entirely resident on the user's computer. Performance considerations will largely dictate the pattern of interaction between the client and the server.
2. A client implemented as a *browser plug-in*. This also can be a fairly rich client, with many of the same kinds of interactions. It is likely that the use of a browser plug-in will allow the implementation of most if not all of the functionality of a rich client executing outside of a browser.
3. A client implemented as one or more *applets* running in a set of web pages. Applets are designed to be as secure as possible, since they are downloaded from a web site to the user's computer. Applets, for example, cannot read or write files on the user's computer, so the user-specific state must be stored on the web site that the applet came from and downloaded by the applet. The necessity of sending all the executable code along with the information may mean that functionality will need to be limited due to the time it takes to download. Applets are seen less often now than they were in the 1990s. Many uses of applets have now been replaced by JavaScript, Flash, and/or Ajax client-side code (see below for descriptions of these techniques).
4. *AJAX* as an acronym for Asynchronous JavaScript and XML, is a recently-coined term for a set of technologies that have been in use for some time. It is similar if not identical to Dynamic HTML (DHTML). The core concept is that the user interface to the application is provided by JavaScript running within the browser and interacting with a web server by sending and receiving XML documents. AJAX eliminates the need to completely replace web pages in response to user input and allows user interaction that can be more like a desktop application than a browser-based application. Building AJAX applications requires the use of multiple development languages and skills. Development toolkits for AJAX-style applications, such as the BackBase[8] toolkit, have become available in recent years. AJAX is useful for building applications that require high levels of interactivity in an environment that permits JavaScript.
5. *Flash*[9] is one of the standard plug-ins that nearly every browser user has downloaded at some time or another. Flash has a bad reputation in some quarters: probably everyone has at one time or another run into web sites that feature pointless Flash animations that take too long to download and delay us from getting the information we are looking for. But Flash also provides a highly interactive user experience, and is being used to produce web applications with a complete user interface. Like AJAX, Flash applications use scripting to implement application behavior, and for communication with the web server.

[8] BackBase, San Mateo, California.

[9] Flash is a Macromedia product that is now owned by Adobe Systems Inc., San Jose, California.

6. A client implemented as a set of *HTML pages*. This is the most limiting form of *web enabled.* With no executable code on the client, all changes to the information presented to the user must be accomplished by regenerating the web page and sending the new version. This can only be accomplished in response to action in the browser. For instance, the user clicks on a button, causing the contents to be sent to the server in a *form*. The server constructs or loads a web page and sends that page as the response. It is not possible for the web server to *push* a new version of the page autonomously unless some other software is installed on the user's computer.

Of these techniques, the only one guaranteed to be able to handle all types of user interaction is the *rich client*. However, this is not what is normally meant by *web enabled*, since the kind of communication that is used between client and server is invisible to the user. In other words, from the user's point view the *rich client* would not seem to have anything to do with the web.

The popular notion of *web enabled* assumes the use of a web browser of some sort. From a development point of view, the preferred approach to building clients that run in a browser would likely be AJAX. But in high-security environments the downloading of JavaScript to a user's browser may not be permitted. In fact, there is little if any risk in running JavaScript. The language does not permit destructive behavior and security holes that existed in older browser versions have long been fixed. However, there is always the possibility that draconian security regimes may be in place in some prospective environments. In a worst case scenario, it would be necessary to fall back to using HTML only pages.

The HTML web page approach is really only suitable for viewing text and pre-defined images. The user would only be able to interact with the client and server by filling out *forms*, submitting them, and receiving the result. An HTML client would be able to view a graphic image such as a map in the kind of objectified geospatial framework that is commonly used by the military to track friendly and enemy forces. However, the graphic environment would not be interactive. For example, the user would not be able to drag and drop objects (e.g., infrastructure objects such as bridges, buildings, routes) from one location to another on the map. By contrast, in an AJAX or Flash-based application such interaction would require some effort on the part of developer, but it could be accomplished.

All forms of web-enabling have consequences for the possible functionality of a system, but some consequences are more severe than others. It is important to know what the end-users expect when they ask for a *web enabled* system, so that they can be made aware of the limitations of each of the alternative implementations.

Enhancing Value to Knowledge Management Programs

Stephen M. Wolfe, Col, USAF, MSC, DBA

Air Force Institute for Operational Health
Brooks City-Base, TX 78259

James D. Whitlock, Lt Col, USAF, MSC, CPHIMS, MPH, MBA

Office of the Surgeon General, United States Air Force
Boling AFB, Washington, DC 20232

David S. Sanchez, Maj, USAFR, MSC, M.S.H.S.

Air Force Reserve Command
Robins AFB, GA 31098

Abstract

This paper addresses the implementation of an advanced search and classification web-service platform that delivers 100% search precision and retrieval, combines conceptual search technology, performs metadata extraction, and automates the classification of enterprise content. A $6 billion global healthcare system with over 60 thousand employees the Air Force Medical Service (AFMS) is responsible for providing healthcare services to over 2.6 million patients annually. In this paper the authors discuss how the AFMS has executed the transformation of relevant information into actionable, situational, and issue specific knowledge.

Integrating leading edge conceptual search and classification technology into significant enterprise level processes, the authors conclude the existence of capabilities that previously were not present. AFMS leadership is now able to rapidly organize explicit content facilitating more effective communication and decision-making; cross-functional operating units are now able to reduce process timelines, utilize untapped resources, and enhance outcome quality; and, organizations within the AFMS are now able to expedite task performance and decision making while advancing individual and group performance.

The paper discuses the imperatives for effective Knowledge Management in a large enterprise where 100% information retrieval precision is critical to achieving mission outcomes. Attention is directed to the combination of capabilities that combine to provide a competitive advantage to existing KM, e-Learning, and decision support systems. These include concept identification based upon Shannon's Information Theory, Probabilistic Latent Semantic Indexing, and Language Stemming.

Introduction and Challenges

In today's fast paced environment, getting the right information to the right person at the right time has become critical to an organization's success. Key to that success is having the ability to harvest knowledge from unstructured information, essentially tapping into its intellectual capital in an effort to enhance the decision making process and to facilitate the creation and realization of creative solutions to collaboration problems.

Over the past few years, the Department of Defense has proactively deployed Knowledge Management (KM) platforms that have been focused on removing what it considers to be the greatest impediment to inter-organizational knowledge sharing – access (Stonebrooke 2003) (Fig. 1). Yet despite spending millions of dollars converting traditional websites to KM Communities where participants subjectively upload content to organizational and functional communities, end-users still find themselves searching for that information that imparts knowledge and effectively enhances the decision making process.

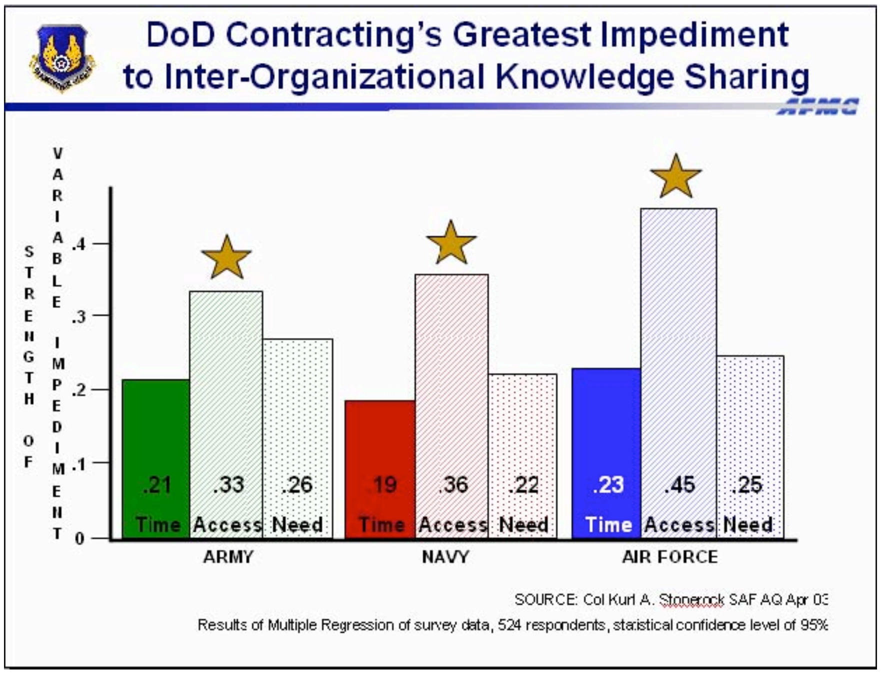


Figure 1

In the Military Healthcare System, each service component's unique Medical Service is responsible for promoting, improving, conserving, and restoring the mental and physical wellbeing of military personnel in an effort to keep its fighting forces healthy and maintain competitive advantage over its enemies (U.S. DoD 2006). Through the efficient management of health service resources, quality preventative medicine, and medical intelligence services, commanders and decision makers at every level of military healthcare are able to obtain an environmental awareness of their unique situations in addition to an enhanced warfighting capability.

Within the U.S. Air Force (USAF), the Office of the Surgeon General has met its obligation thru the proactive deployment of a variety of tools that are focused on providing support to decision makers. Fundamental to the AFMS solution is an integrated approach that automates the time consuming process of collecting, indexing, categorizing and classifying unstructured enterprise information from discrete information sources which results in the rapid delivery of actionable information to end-users, and the leveraging and increased utilization of existing AFMS technologies.

In order to bring value to its existing KM program, the AFMS realized it must be able to bring coherency to isolated document sets within its domain – a domain that includes 270 geographically dislocated Air Force medical units, 74 military treatment facilities, and a population of over 60,000 healthcare professionals. This effort began with the AFMS intranet site known as the Knowledge Exchange (Kx) (Fig. 2). Designed to store, share, and exchange knowledge among AFMS members the Kx contains over 350 distinct sites arranged in a hierarchical manner and organized by formal organization function (Whitlock 2007a).

Converting technological components of the AFMS Kx portal to enhance knowledge discovery efforts for documents that had been subjectively loaded onto the Kx proved to be a straight forward process. How leadership was going to capture knowledge from unstructured information residing on over 270 different networks and incorporate that information into the decision making process of enterprise business processes at the strategic level was our greatest challenge.

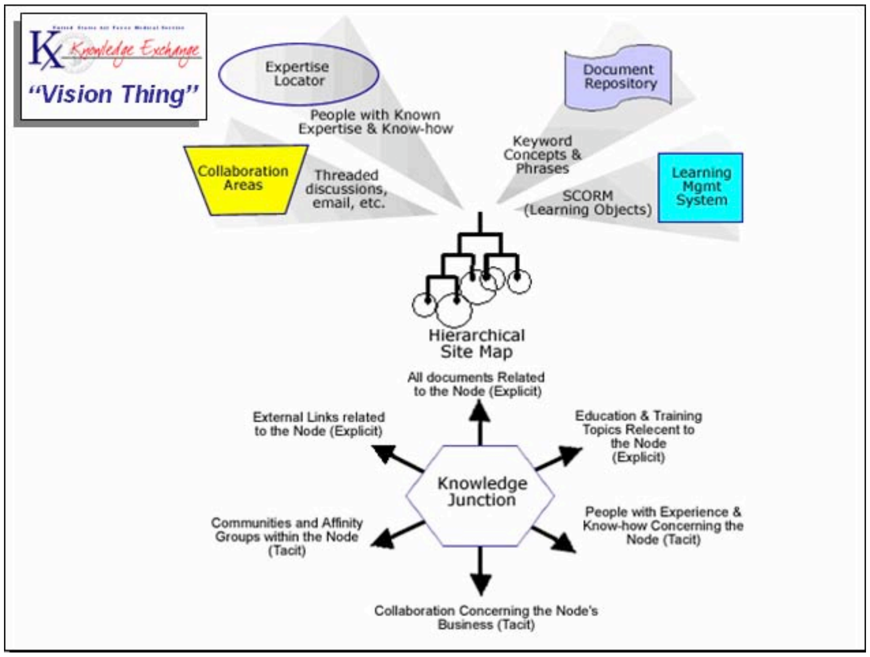

Kx Knowledge Exchange
"Vision Thing"
Expertise Locator
People with Known Expertise & Know-how
Document Repository
Keyword Concepts & Phrases
Collaboration Areas
Threaded discussions, email, etc.
Learning Mgmt System
SCORM (Learning Objects)
Hierarchical Site Map
All documents Related to the Node (Explicit)
External Links related to the Node (Explicit)
Education & Training Topics Relecent to the Node (Explicit)
Knowledge Junction
Communities and Affinity Groups within the Node (Tacit)
People with Experience & Know-how Concerning the Node (Tacit)
Collaboration Concerning the Node's Business (Tacit)

Figure 2

Imperative for Natural Language Knowledge Discovery

The content resident within the Air Force enterprise provides a wealth of information and potential knowledge but it lacks structure, and accessibility is limited. This problem is compounded by the following realities:

- Paper-based manual processes lead to lost information and lack of knowledge sharing

- No standard method for knowledge discovery across the enterprise makes finding information labor intensive across all levels of involvement
- Manual enterprise business processes are not all-inclusive – Active/Reserve/Guard units
- Staff moves frequently – from base to base or job to job within same organization
- Some parts of our mission are common to the private sector, many are not
- Critical need exists for centralized policy coordination and guidance
- Precision of existing knowledge discovery service does not meet the level required

As a result of their frustration at not being able to find information, many AFMS users resort to Google in an effort to locate information that may facilitate their decision making process. Google's efficient ranking of search results is based on relationships (links) between content (webpages). Because Google's ranking is effective, many users don't venture beyond the first search results page. While this looks good on the surface, Google's magic does not work in enterprise search because similar relationships do not exist between enterprise content. Enterprise search users get frustrated when they do not see high precision in search results so they assume that relevant content does not exist (Whitlock 2007b).

Like Google most information retrieval systems provide either High Precision – Low Recall or Low Precision – High Recall. In the former, end-users may place phrases in quotes to receive 100% precision however, the result will only include exact matches to the query phrase and will not take into account concepts that the phrase may imply. In the latter, keywords are used and the result set is so large that end users may only look at the top 5 results on the first page and make the false determination that anything beyond the first page of results is of no value. In the end, both types of results fail to provide the user with the most relevant information all the time, or what we will call High Precision - High Recall.

Key Differentiators

Achieving High Precision – High Recall is essential to leveraging explicit intellectual capital, and implicit knowledge and know-how. The AFMS has accomplished this by placing compound terms into an index, implementing a weighting schema for compound terms, and relevance ranking compound terms at index time. At the technical level, this involved applying aspects of Shannon's Information Theory, Probabilistic Latent Semantic Indexing (PLSI), and Language Stemming.

The AFMS implementation of Concept Searching technology interprets Shannon's Information Theory in an Information Retrieval context to compute the incremental value of a compound term over its single-word components. Higher order compound terms are evaluated using their lower order compound components (Challis 2006a).

For example, if we have the concept of "heart surgery" then we compute the incremental value (how much extra information) of this two-word concept over and above its two single-word components. For a three-word concept like: "open heart surgery" we would look for the incremental value of the complete term over and above "open heart" and "heart surgery". The results are totally dependent upon the totality of documents being indexed since this is an adaptive technology that automatically tunes itself based on the documents it is given.

It is no coincidence that the majority of compound terms are proper nouns, noun phrases and verb phrases. These sentence fragments convey the key concepts in most text. By correctly weighting these compound terms Concept Searching is able to identify documents containing concepts the user is looking for rather than finding documents that contain the right words.

Probabilistic Latent Semantic Indexing (PLSI) enables a user to retrieve documents that are relevant to a query even when the query does not contain words used in the query test. It also ignores documents within repositories that do contain words from the user's query but are not relevant to the user's query (Challis 2006b). PLSI is achieved by:

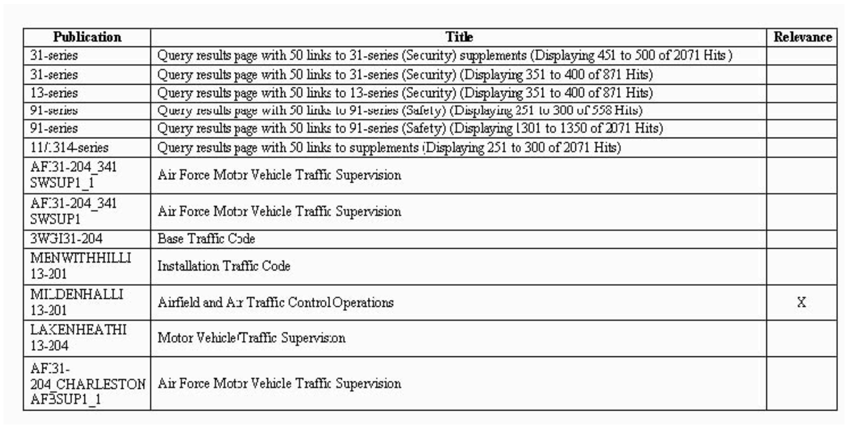
- Relevance ranking the documents matched by the initial query
- Extracting the distinguishing concepts from the most relevant documents
- Expanding the query to include selected related concepts

Imagine a query that asks "do caskets need to be pressurized" and finding documents related to the "transportation of contaminated human remains" and "mortuary affairs" and some of the retrieved documents do not contain any words from the original query. That is PLSI.

Often a user types in a query with one form of a word but would like to match other forms of what is essentially the same word. By implementing a suffix-stripping algorithm, Information Retrieval systems are able to normalize words such as "swims" with "swim", "swims", "swimmers", and "swimming".

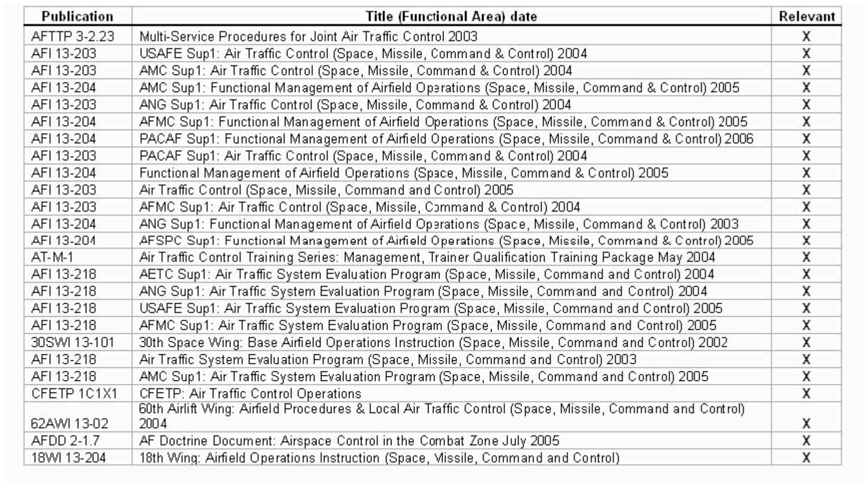
Language stemming also accommodates users who may accidentally misspell a term within a query: "tripple" is matched with "triple" and both "comission" and "commision" are matched with "commission".

To demonstrate a High Precision – High Recall capability we will look at how 2 different information retrieval (IR) applications provide end users with information around a specific concept. In this scenario over 30,000 documents from the Air Force Electronic Publications Library were indexed on two different IR platforms. The end user then solicits both applications for information around the concept of air traffic. The first platform provides a Low Precision – High Recall result where the end user must open 305 results until there is any content related to the concept of air traffic (Fig. 3). When the same concept query is placed into the second platform a High-Precision – High Recall results set appears (Fig. 4).



Publication	Title	Relevance
31-series	Query results page with 50 links to 31-series (Security) supplements (Displaying 451 to 500 of 2071 Hits)	
31-series	Query results page with 50 links to 31-series (Security) (Displaying 351 to 400 of 871 Hits)	
13-series	Query results page with 50 links to 13-series (Security) (Displaying 351 to 400 of 871 Hits)	
91-series	Query results page with 50 links to 91-series (Safety) (Displaying 251 to 300 of 558 Hits)	
91-series	Query results page with 50 links to 91-series (Safety) (Displaying 1301 to 1350 of 2071 Hits)	
11/:314-series	Query results page with 50 links to supplements (Displaying 251 to 300 of 2071 Hits)	
AF:31-204_341 SWSUP1_1	Air Force Motor Vehicle Traffic Supervision	
AF:31-204_341 SWSUP1	Air Force Motor Vehicle Traffic Supervision	
3WGI31-204	Base Traffic Code	
MENWITHHILLI 13-201	Installation Traffic Code	
MILDENHALLI 13-201	Airfield and Air Traffic Control Operations	X
LAKENHEATHI 13-204	Motor Vehicle/Traffic Supervision	
AF:31-204_CHARLESTON AF3SUP1_1	Air Force Motor Vehicle Traffic Supervision	

Figure 3



Publication	Title (Functional Area) date	Relevant
AFTTP 3-2.23	Multi-Service Procedures for Joint Air Traffic Control 2003	X
AFI 13-203	USAFE Sup1: Air Traffic Control (Space, Missile, Command & Control) 2004	X
AFI 13-203	AMC Sup1: Air Traffic Control (Space, Missile, Command & Control) 2004	X
AFI 13-204	AMC Sup1: Functional Management of Airfield Operations (Space, Missile, Command & Control) 2005	X
AFI 13-203	ANG Sup1: Air Traffic Control (Space, Missile, Command & Control) 2004	X
AFI 13-204	AFMC Sup1: Functional Management of Airfield Operations (Space, Missile, Command & Control) 2005	X
AFI 13-204	PACAF Sup1: Functional Management of Airfield Operations (Space, Missile, Command & Control) 2006	X
AFI 13-203	PACAF Sup1: Air Traffic Control (Space, Missile, Command & Control) 2004	X
AFI 13-204	Functional Management of Airfield Operations (Space, Missile, Command & Control) 2005	X
AFI 13-203	Air Traffic Control (Space, Missile, Command and Control) 2005	X
AFI 13-203	AFMC Sup1: Air Traffic Control (Space, Missile, Command & Control) 2004	X
AFI 13-204	ANG Sup1: Functional Management of Airfield Operations (Space, Missile, Command & Control) 2003	X
AFI 13-204	AFSPC Sup1: Functional Management of Airfield Operations (Space, Missile, Command & Control) 2005	X
AT-M-1	Air Traffic Control Training Series: Management, Trainer Qualification Training Package May 2004	X
AFI 13-218	AETC Sup1: Air Traffic System Evaluation Program (Space, Missile, Command and Control) 2004	X
AFI 13-218	ANG Sup1: Air Traffic System Evaluation Program (Space, Missile, Command and Control) 2004	X
AFI 13-218	USAFE Sup1: Air Traffic System Evaluation Program (Space, Missile, Command and Control) 2005	X
AFI 13-218	AFMC Sup1: Air Traffic System Evaluation Program (Space, Missile, Command and Control) 2005	X
30SWI 13-101	30th Space Wing: Base Airfield Operations Instruction (Space, Missile, Command and Control) 2002	X
AFI 13-218	Air Traffic System Evaluation Program (Space, Missile, Command and Control) 2003	X
AFI 13-218	AMC Sup1: Air Traffic System Evaluation Program (Space, Missile, Command and Control) 2005	X
CFETP 1C1X1	CFETP: Air Traffic Control Operations	X
62AWI 13-02	60th Airlift Wing: Airfield Procedures & Local Air Traffic Control (Space, Missile, Command and Control) 2004	X
AFDD 2-1.7	AF Doctrine Document: Airspace Control in the Combat Zone July 2005	X
18WI 13-204	18th Wing: Airfield Operations Instruction (Space, Missile, Command and Control)	X

Figure 4

Automated Classification

Existing document management systems are heavily dependent on subject matter experts who upload and organize information within their respective communities. This is both time consuming and subjective, resulting in a stove-piped, static presentation of information. Deploying an automated classification capability within a medical wing and a KM portal entailed the creation of unique terms that would enable the classification of unstructured information to ontologies that were aligned to AFMS organizational units, functional medical specialties, and unique medical warfighting missions (Fig. 5). As the foundation for this program, a total of 8,649 terms/"clues" were created. When contained within an unstructured document and when meeting a pre-determined level of relevance in that document, these clues serve as a set of rules that direct the system to classify documents to single and multiple locations at a rate of 200,000 documents per hour (Challis 2006c).

At the 311th Human Systems Wing over 658,000 unstructured documents were automatically collected, indexed, and classified into organizational, functional, and unique wartime mission taxonomies. Located on organizational, division, and branch level shared drives these documents reflected norms, values, expertise, and process outcomes and were in a sense, less tangible than formal risk communication contained within published scientific and technical information reports. If certain documents contained content relating to multiple levels within or across taxonomies, those documents were classified to those levels automatically and objectively while occupying the space of a single copy of the document as opposed to the multiple amounts of space that would be required if a person individually filed that same document in multiple locations (Fig. 6).

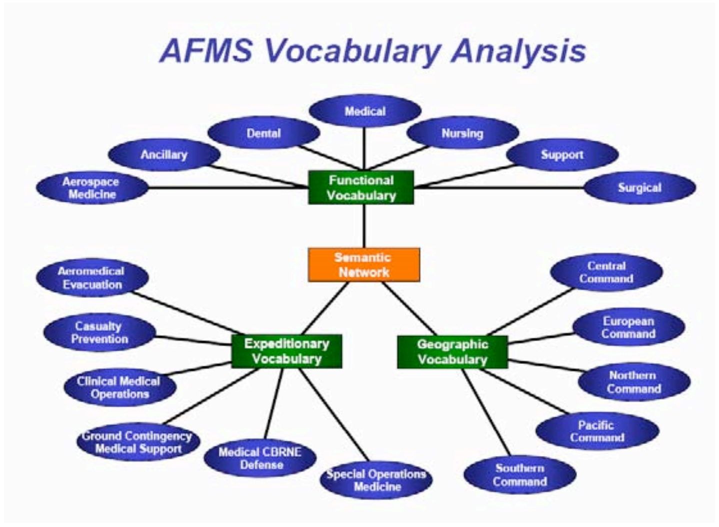
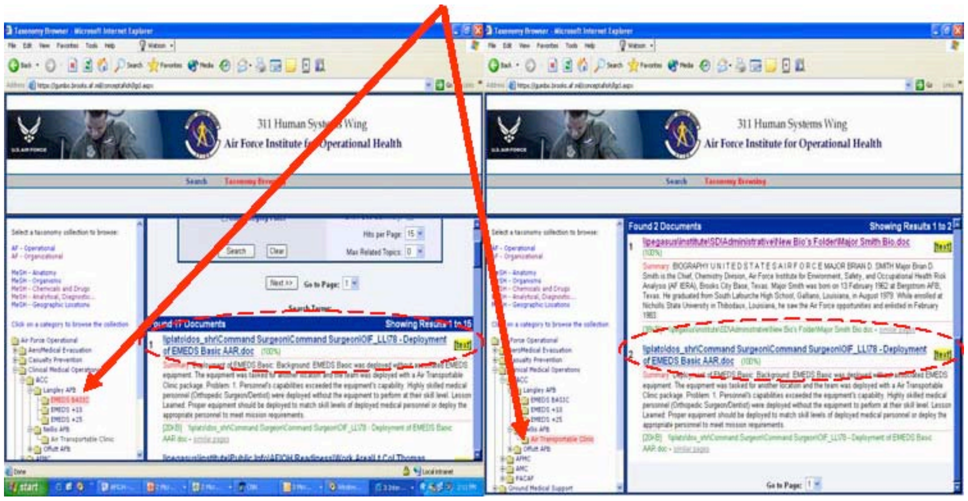


Figure 5

Results: Same document automatically classified into two separate applicable areas – in this case Unit Type Codes at pilot units within a Major Command and accountable to the Clinical Medical Operations Functional Area Working Group



Found 17 Documents — Showing Results 1 to 15

1 \\plato\dos_shr\Command Surgeon\Command Surgeon\OIF_LL78 - Deployment of EMEDS Basic AAR.doc (100%) [text]

Summary: Deployment of EMEDS Basic: Background: EMEDS Basic was deployed without associated EMEDS equipment. The equipment was tasked for another location and the team was deployed with a Air Transportable Clinic package. Problem: 1. Personnel's capabilities exceeded the equipment's capability. Highly skilled medical personnel (Orthopedic Surgeon/Dentist) were deployed without the equipment to perform at their skill level. Lesson Learned: Proper equipment should be deployed to match skill levels of deployed medical personnel or deploy the appropriate personnel to meet mission requirements.

[20KB] \\plato\dos_shr\Command Surgeon\Command Surgeon\OIF_LL78 - Deployment of EMEDS Basic AAR.doc - similar pages

Found 2 Documents — Showing Results 1 to 2

1 \\pegasus\Institute\SD\Administrative\New Bio's Folder\Major Smith Bio.doc (100%) [text]

Summary: BIOGRAPHY U N I T E D S T A T E S A I R F O R C E MAJOR BRIAN D. SMITH Major Brian D. Smith is the Chief, Chemistry Division, Air Force Institute for Environment, Safety, and Occupational Health Risk Analysis (AF IERA), Brooks City Base, Texas. Major Smith was born on 13 February 1962 at Bergstrom AFB, Texas. He graduated from South Lafourche High School, Galliano, Louisiana, in August 1979. While enrolled at Nicholls State University in Thibodaux, Louisiana, he saw the Air Force opportunities and enlisted in February 1983.

\\pegasus\Institute\SD\Administrative\New Bio's Folder\Major Smith Bio.doc - similar pages

2 \\plato\dos_shr\Command Surgeon\Command Surgeon\OIF_LL78 - Deployment of EMEDS Basic AAR.doc (100%) [text]

Summary: Deployment of EMEDS Basic: Background: EMEDS Basic was deployed without associated EMEDS equipment. The equipment was tasked for another location and the team was deployed with a Air Transportable Clinic package. Problem: 1. Personnel's capabilities exceeded the equipment's capability. Highly skilled medical personnel (Orthopedic Surgeon/Dentist) were deployed without the equipment to perform at their skill level. Lesson Learned: Proper equipment should be deployed to match skill levels of deployed medical personnel or deploy the appropriate personnel to meet mission requirements.

[20KB] \\plato\dos_shr\Command Surgeon\Command Surgeon\OIF_LL78 - Deployment of EMEDS Basic AAR.doc - similar pages

Figure 6

Lessons Learned Process

Lessons learned come from after-action reports. After-action reports are written by all Air Force elements that participate in exercises and real world operations. They also apply to humanitarian, base closure, peacekeeping, and noncombatant evacuation operations. The purpose of after-action reporting is to identify issues that arose during the above, good or bad, and task mitigation to an appropriate authority for action.

Embedded within most after-action reports are techniques, procedures, or practical work-arounds that enabled a task to be accomplished to acceptable standards based upon an identified deficiency or shortcoming. This embedded content is collectively referred to as 'lessons learned'. Once a 'lesson learned' is identified it enters a process whereby it is translated into an action that corrects a deficiency; operational performance is the motivation, with timeliness and accuracy being key success factors.

In October 2004, the Air Force Institute for Operational Health (AFIOH) and the Air Force Medical Support Agency (AFMSA) KM Program Manager initiated efforts to develop a functional Concept Model to demonstrate automated collection, indexing, and classification of after-action reports against functional, expeditionary, and organizational taxonomies. The purpose, in part, was curiosity of the reported benefit of an automated collection, indexing, and classification methodology and having a benchmark to compare results against. To test the Concept Model, AFIOH compared a conventional process whereby after-action reports from Operations ENDURING FREEDOM and IRAQI FREEDOM were collected, indexed, and classified with the automated Concept Model (Wolfe 2005).

The conventional (manual) process began with 9 officers, each from different medical career fields, reading groups of reports with the objective of placing each into one of several common information categories. This process took 4 months and involved countless reviews and meetings to resolve differences in perception; each report had a different format, and many were either inappropriately tasked to a wrong location or incompletely tasked due to lack of subject matter expertise of a particular reviewer.

The Concept Model was designed to facilitate the automated process. AFIOH created an expeditionary ontology consisting of classes of terms and class clues that were developed based on content from the National Library of Medicine Medical Subject Headings (MESH) database and the Air Force Medical Logistics Office (AFMLO) allowance standard database.

Using this "objective" taxonomy from a validated source resulted in AFIOH collecting, indexing, and classifying 208 after action reports in just 28 seconds (We did it twice to be sure)! Instead of creating a folder/file structure and dragging and dropping copies of the same document to many folders based on an individual's limited perspective, the use of classes and class clues from ontologies allowed AFIOH to rapidly classify each after-action report into their appropriate areas for action. Many times one report would touch on 3 to 4 subject areas. Each report was linked to multiple folders, ensuring that the information was reaching its appropriate destination for action.

The successful demonstration of applying an automated indexing and classification capability to a real world process did not occur overnight. While the teams of 9 officers created their own classification vocabularies as they reviewed their after-action reports, the concept model team created an ontology composed of controlled MESH and AFMLO vocabularies. In using controlled vocabularies the team was able to objectively describe concepts and relationships that exist for the Expeditionary AFMS community without having to read after-action reports and subjectively establish a structure for information classification. Once information was placed into an index it was classified to its appropriate location for action. Today, information loaded onto the AFMS Kx is immediately incorporated into the existing index and classified to its appropriate folder and knowledge junction based upon an AFMS ontology comprised of over 9,000 weighted classes and clues whose origination was in the form of a controlled vocabulary.

Accelerating task performance by reducing process timelines has provided Air Force leadership with an opportunity to apply proven retrieval and classification techniques to other enterprise processes. Our next objective focused on the acquisition process, a $210 billion Air Force investment strategy used to acquire systems and services that provide combat capability to joint warfighting commanders.

Human Systems Integration and Mission Capability Gap Process

Human Systems Integration (HSI) is a comprehensive strategy used early in an acquisition process to optimize total system performance, minimize total ownership costs, and ensure that the system is built to accommodate the characteristics of the user population that will operate, maintain, and support the system (Fig. 7).

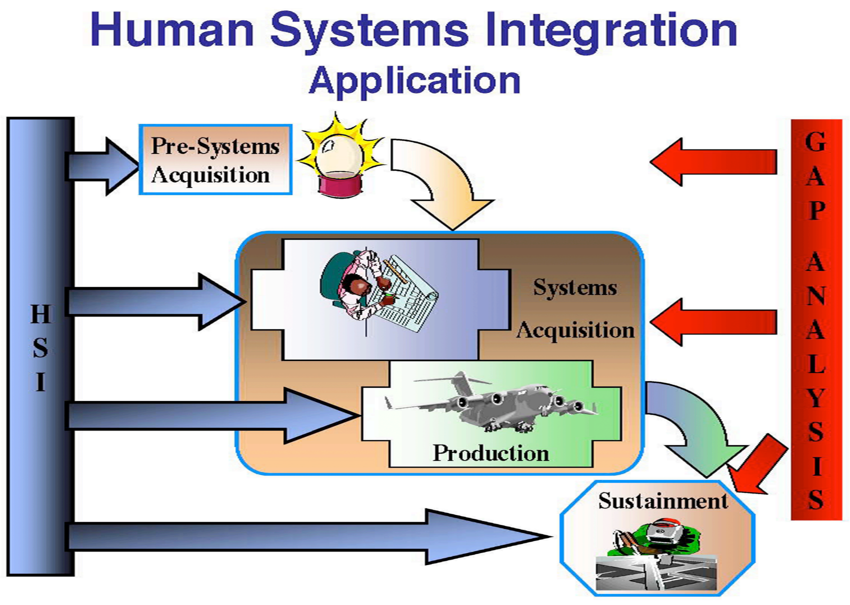


Figure 7

HSI is not one single thing but rather a process or strategy that must be applied for systems to meet the needs of the mission as well as the user, supporter, and maintainer. HSI is applied at all stages of the deployment of a weapon system. Ideally, and in order to save the most money, HSI is most usefully applied during the concept development phase to ensure that the system reaches the front line in a useable and supported form. While this process is rather straight forward for new weapon systems, reality presents the Air Force with legacy systems. This is where the capability gap occurs and the need for effective analyses materializes.

The Capability Gap Analysis program functions to identify gaps in mission capability due to human performance deficiencies and highlights them for solution. When a solution at the local level is not possible, the analysis is passed thru the Human Performance Wing to higher headquarters for evaluation, prioritization, and eventual resolution. At the local level, Aerospace Medicine physicians submit mission capability gap analysis findings resulting from assessments conducted in the course of routine mission participation and health safety inspections (Chapple and Surman 2007).

Prior to the implementation of the AFMS Knowledge Discovery capabilities, Aerospace Medicine physicians identified mission capability gaps as they became evident. However, due to limited information retrieval capability across multiple information domains, mission capability gaps were often identified and resolved in isolation. End-users researching solutions to real-world problems were forced to interrogate multiple information management platforms; conceptual relationships between content residing at multiple locations would have to be made in a subjective manner based upon the end-users experience (Fig. 8). Subsequently, the Air Force was not as effective as it could be in identifying mission capability gaps, communicating gaps that negatively influence performance, resolving gaps, and positively influencing the acquisition process.

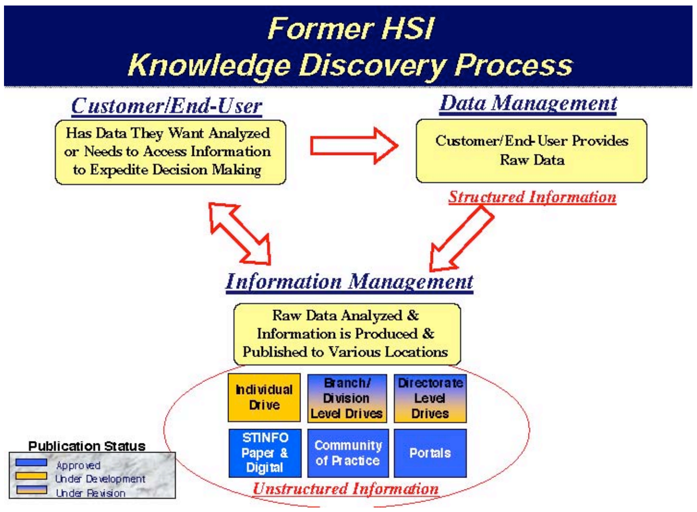


Figure 8

With the deployment of concept searching and classification capability within the AFMS, the Human Performance Wing is in the process of automating the capability gap analysis process across Active Duty, Air National Guard, and Air Force Reserve flying and maintenance units. By classifying unstructured information into seven HSI domains (Fig. 9) unstructured information is placed into actionable context (Fig. 10). Aerospace Medicine providers can now interact with a single source that has objectively extracted information on enterprise servers and distilled that information into factors that effect human performance; they can then initiate steps to resolve specific mission capability gaps.

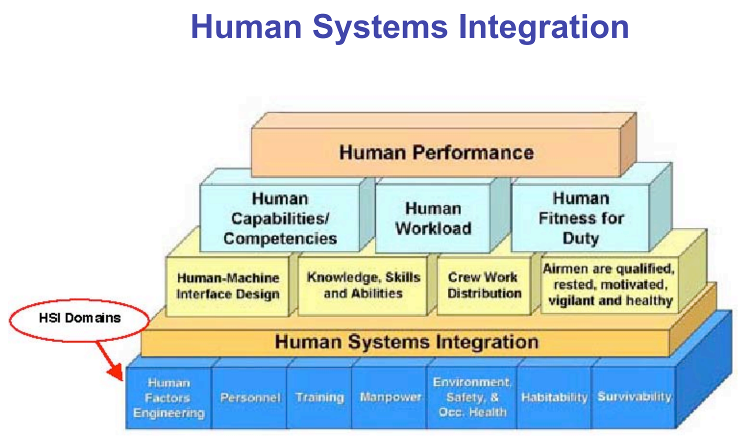

Human Systems Integration
Human Performance
Human Capabilities/ Competencies
Human Workload
Human Fitness for Duty
Human-Machine Interface Design
Knowledge, Skills and Abilities
Crew Work Distribution
Airmen are qualified, rested, motivated, vigilant and healthy
HSI Domains
Human Systems Integration
Human Factors Engineering
Personnel
Training
Manpower
Environment Safety, & Occ. Health
Habitability
Survivability

Figure 9

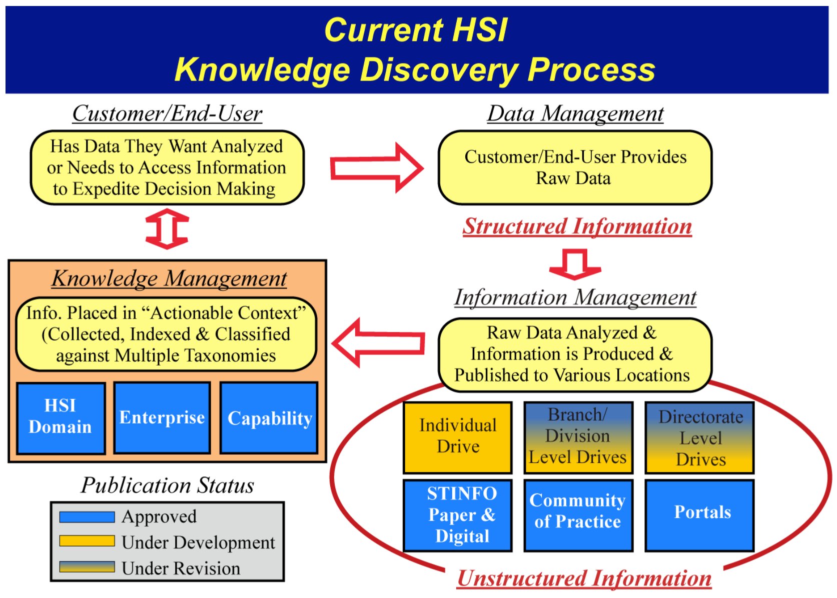

Current HSI
Knowledge Discovery Process
Customer/End-User
Has Data They Want Analyzed or Needs to Access Information to Expedite Decision Making
Data Management
Customer/End-User Provides Raw Data
Structured Information
Information Management
Raw Data Analyzed & Information is Produced & Published to Various Locations
Knowledge Management
Info. Placed in "Actionable Context" (Collected, Indexed & Classified against Multiple Taxonomies
HSI Domain
Enterprise
Capability
Individual Drive
Branch/ Division Level Drives
Directorate Level Drives
STINFO Paper & Digital
Community of Practice
Portals
Publication Status
Approved
Under Development
Under Revision
Unstructured Information

Figure 10

Discussion

Customer, staff, and organizational needs, internal perceptions and motivations, strategies, relevant knowledge and awareness of human and financial capital resources combine to serve as the foundation from which raw information is transformed into something that is actionable within a business operation. Connecting decision makers with relevant and timely information in an efficient manner can have a direct and positive impact not just for a particular operation but also a specific outcome.

Creating and maintaining access for end-users to separate databases and information communication platforms results in limited end-user access and requires extensively trained administrative and technical resources. Since the content of unstructured business operations and customer information is critically linked to nearly every decision making process across a wide range of operations, organizations must be able to efficiently process their unstructured information (Evolvent 2005).
Thru the transformation of relevant information into actionable knowledge, the AFMS has realized three significant benefits. For leadership, they are able to rapidly organize their organization's explicit and implicit content to facilitate more effective communication and decision-making. For staff, cross-functional operating units are able to push relevant information to interested persons, reduce process timeline, utilize untapped resources, and enhance outcome quality. For the organization, it avails contemporary and relevant information that assists and expedites task performance and decision making advancing individual and group performance via enhanced situational and issue-specific knowledge.

Leveraging existing technology platforms and information resources within the AFMS is resulting in increased utilization of its information resources. Thru an incremental approach senior leadership is able to obtain a significant return on investment in a relatively short amount of time as demonstrated by the rapid improvement in support of information discovery and communication of healthcare operations related information across organizational and functional lines.

References

Challis J. (2006); Concept Searching Technology Overview; www.ConceptSearching.com.

Chapple S. and C. Surman; Human Systems Integration; Performance Enhancement Directorate, 311th Human Systems Wing, Brooks City-Base, TX, February 2007.

Evolvent; Knowledge Management Discovery Toolset; Evolvent Magazine, Fall: 25-30, 2005.

Stonebrook K; Acquisitions Multiple Regression Survey; Office of the Assistant Secretary of the Air Force for Acquisitions, April 2003.

U.S. Department of Defense (2006); Force Health Protection, Office of the Assistant Secretary of Defense for Health Affairs.

Whitlock J.D.; The Future of Search Technologies; The 8th Annual Knowledge Management Conference and Exhibition: E-Gov Institute, Washington, D.C., April 3-5, 2007.

Wolfe S.; Next Steps in Content Management: Concept Searching; Evolvent Magazine, Summer: 21-22, 2005.

Context-Based Information Systems Development

Bogdan D. Czejdo[1] , Thompson Cummings[1] , Mikolaj Baszun[2]

[1] St. George's University, Grenada
[2] Warsaw University of Technology, Poland

Abstract

Information System development involves processing a large number of documents containing a substantial amount of domain knowledge. Most of these documents are written in natural language and they usually contain ambiguity, contradictions and redundancy. These documents are often incomplete. The background knowledge or context knowledge can help resolve these problems. Additionally, important application of the background knowledge is that it helps in "fact mining" for facts that can be omitted during information system development. We propose to build and use an external ontology representing background knowledge. Guided by this ontology the text mining techniques can be used to extract domain knowledge from provided documents. We use an extended Unified Modeling Language (UML) for graphical representation of knowledge modeling. Such knowledge includes not only "directly implementable" knowledge that can be converted into traditional database schema, but also general constraints that should be taken into consideration while developing information systems.

1. Introduction

Information System development involves processing a large number of documents containing a substantial amount of domain knowledge. Most of these documents are written in natural language and they usually contain ambiguity, contradictions and redundancy. These documents are often incomplete. The background knowledge or context knowledge can help resolve these problems. Additional important application of the background knowledge is that it helps in "fact mining" for facts that can be omitted during information system development. We propose to build and use an external ontology representing background knowledge. Guided by this ontology the text mining techniques can be used to extract domain knowledge from provided documents. We use an extended Unified Modeling Language (UML) for graphical representation of knowledge modeling. Such knowledge includes not only "directly implementable" knowledge that can be converted into traditional database schema, but also general constraints that should be taken into consideration while developing information systems.

A variety of approaches were used to store, process, and query the knowledge [1, 2, 3, 5, 10, 14, 15, 16, 17, 18, 22, 23, 24, 25]. In this paper, we will use the ontology to capture background knowledge and Unified Modeling Language (UML) for domain knowledge modeling. UML was designed for visualizing, specifying, constructing, and documenting the typical artifacts of software systems [4, 11, 12, 13, 19, 20, 21]. We showed in several papers [6, 7, 8, 9] how we can go beyond traditional software modeling and use UML diagrams to capture knowledge about a variety of subjects. This extension was crucial for the application of background knowledge to semi-automatic information system development

Current techniques for knowledge extraction from text are either keyword/category based or structure dependent. Our approach to text mining is to use the combination of qualitative and quantitative techniques for identification of relevant concepts and relationships. The selected concepts and relationships are extracted in the process guided by ontologies for the domain of interest.

In this paper we describe Context-based Processing system as shown in Fig. 1. Our approach assumes multi-stage processing. These stages include ontology processing, knowledge discovery from documents, queries for ontology and documents, and generation/assistance in information system design.

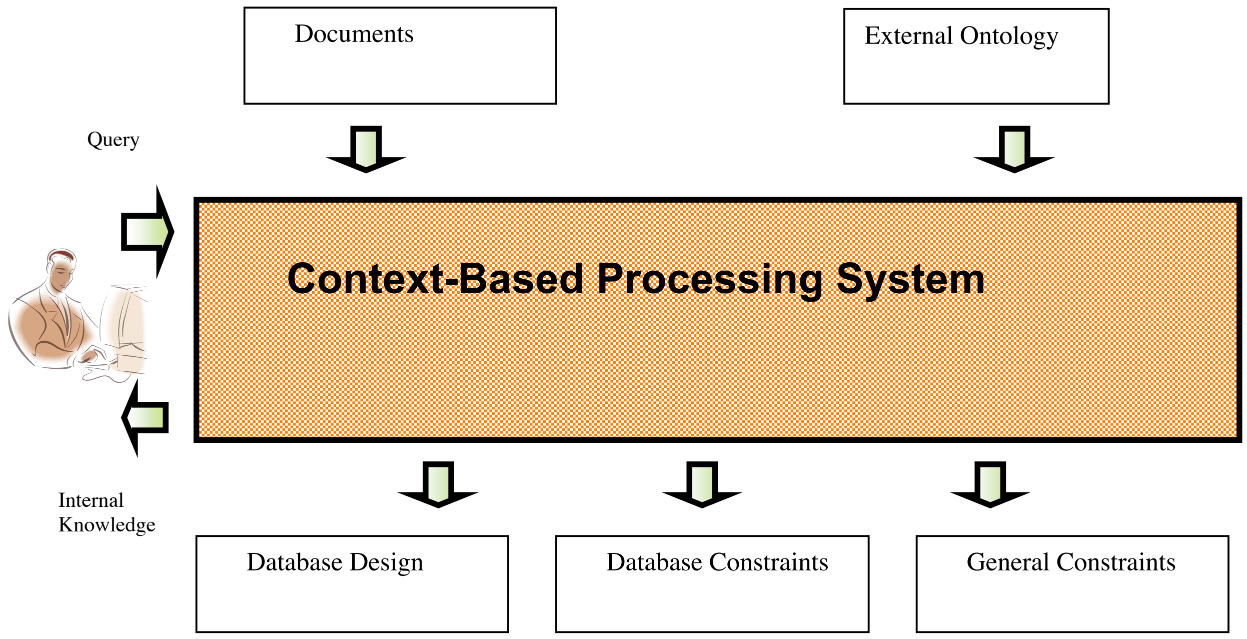


Figure 1: An Overview of Context-Based Processing System

2. Context-Based Processing System

Whenever humans process an individual document (text) they view it in a broad context of facts and rules acquired throughout our life. Let us refer to this background context as a background knowledge or simply context. This background knowledge can be in various forms. Let us assume that it is in the form of ontology containing some digested information such as dictionaries. In general, the context ontology can contain not only the list of concepts as in simple dictionary but also explicit concept hierarchy and relationships between concepts.

Our understanding of a new document very much depends on the "scope" of our background knowledge. Various scopes of background knowledge and "quality" of background knowledge can be different for different areas. One of the measures of ontology "quality" is ontology completeness levels. By completeness level, we mean an overall completeness level or completeness level with respect to a subset of documents or even a completeness level with respect to single document or its fragment. This is an important measure of how "good" ontology is for the specific set of documents. When several distinct sets are processed the completeness measure for each subset may need to be calculated.

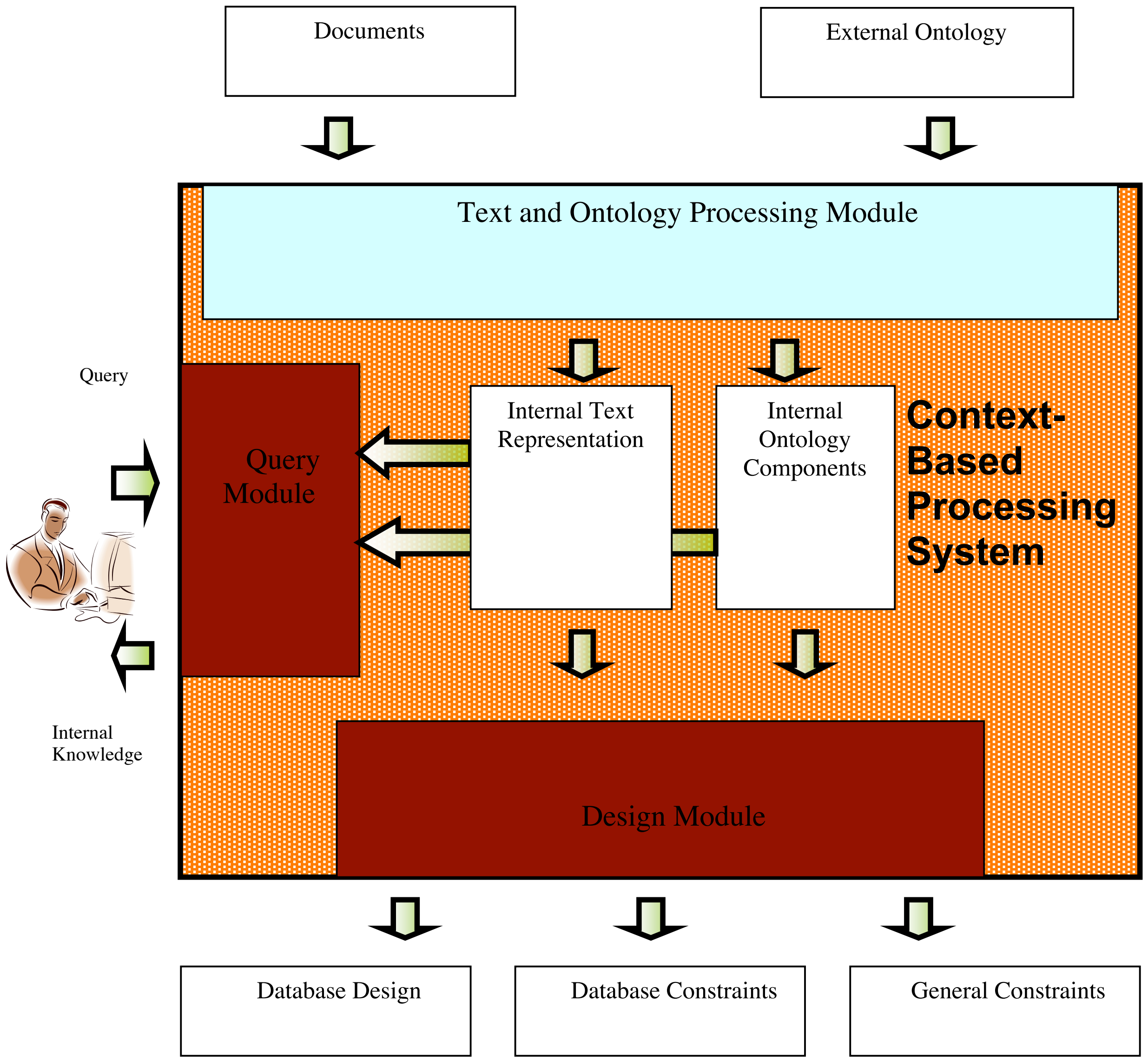


Figure 2: An Overview of a Context-Based Processing System

Our context-based processing system is flexible in the sense that it can be constructed from various available modules. Each module is highly parametric allowing us to adjust parameters for the existing needs. The architecture of our context-based processing system is shown in Fig 2. It is constructed from relatively independent modules: Text and Ontology Processing Module, Query Module and Design Module.

Internal document representation can include concepts, hierarchy of concepts and concepts relationships. For each representation level we can apply various statistical processing types, for example frequencies of words, direct collocations, and distant collocations. For each pair of representation level and processing type, we can use various internal representation components:

simple and compound concepts, relationships, word stems, synonyms, antonyms, stop words, stop modifiers, etc. Not all combinations of system parameter values are important to consider. Some combinations of parameter values are useful in specific situations only.

Based on internal representation, the query can be issued or discrepancies identified. The query can be: (a) keyword search, (b) hierarchy and relationship search, or (c) comparative. The database design procedures are performed after all discrepancies are identified.

3. Text and Ontology Processing Module

Context-based processing is done in several stages. In the first stage, the internal ontology is created by using components of external ontology (if available) and statistical processing of documents as shown in Fig. 3. In this stage, relevant concepts, hierarchy of concepts and relationships between concepts are identified in documents and represented internally.

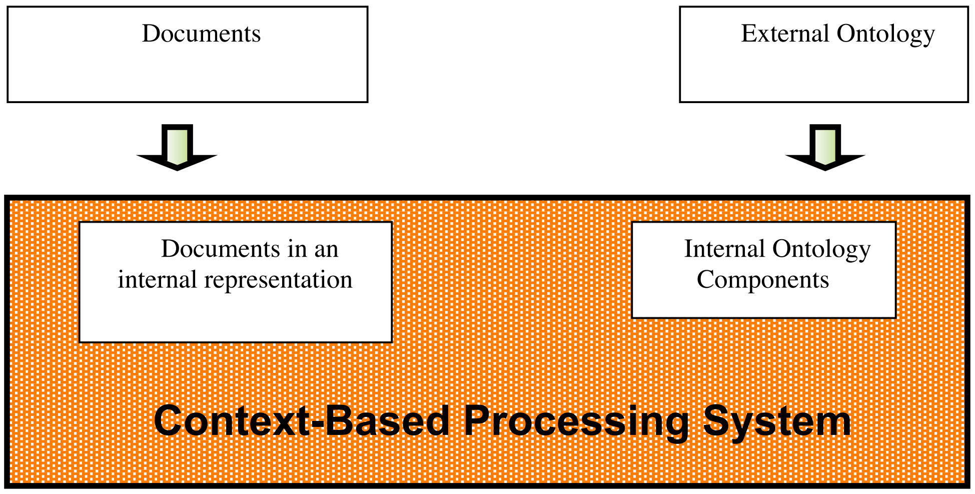

Documents
External Ontology
Documents in an internal representation
Internal Ontology Components
Context-Based Processing System

Figure 3: Text and Ontology Processing

There are different requirement for internal ontology components depending on text recognition levels. For the lexical text recognition level, the ontology should contain the list of concepts, their possible representations and the methods to identify them. Specifically, an ontology for lexical recognition can contain: (a) list of concepts represented by primary names, alternative names, stem words, and synonyms, (b) list of stop words, and if applicable (c) list of misspellings. For the list of concepts, we assume that each concept is represented by a unique primary name, e.g. "**Student",** that we call *name*. Each concept *name* is associated with a list of alternative names, e.g. "**students,"** to allow for alternative forms, singular and plural, etc. The list of alternative words can be also generated from the stem words, e.g. assuming a stem word "**course,"** the alternative name "**courses"** can be generated by appending the "s" to the stem word. In general, both generated and stored alternative words can be used. Each concept has a list of synonyms, e.g. "**professor"** for the concept "**Faculty."** Each concept can have a list of misspelled words. The list of misspelled words can be also generated from the alternative names by using some rule e.g. removing one character, replacing one letter by another letter. In general, both generated and stored misspelled words can be used.

Guided by context ontology the text extraction process can be accomplished. Let us assume a simple paragraph to translate:

> *SEA Online College has many students and many faculty. Each student has a ssn, name and current telephone number. Each faculty has a unique name and office number. Each faculty teaches 3 courses. Students send exams to faculty. Faculty corrects exams and computes grades. Next he sends grade to the gradebook. Gradebook has grade for each course and for each student. Students can see only their own grades in the gradebook.*

Our analysis of the sentences of the paragraph sought first to identify concepts from the text existing in the external ontology. These concepts are loaded to the internal ontology together with relevant properties and relationships. For our example let us assume that all concepts in bold were found in ontology:

> *SEA Online **College** has many **students** and many **faculty**. Each **student** has a **ssn, name** and current **telephone number**. Each **faculty** has a **unique name** and **office number**. Each **faculty** teaches **3 courses**. **Students** send exams to **faculty**. Faculty corrects exams and computes **grades**. Next he sends **grade** to the **gradebook**. **Gradebook** has **grade** for each **course** and for each **student**. **Students** can see only their own **grades** in the **gradebook**.*

Let us also assume that some of these concepts are identified in ontology as typical classes or attributes e.g. **College**, **Student**, **Faculty** are identified as typical classes and **ssn**, **name** and **telephone number** are identified as properties.

Our analysis of the sentences of the paragraph sought to identify the tuples consisting of the following categories: Object 1—Relationship—Object 2. That analysis need to be done in phases where in the first phase we allow for complex objects or object information. The ontology again is the guide to discover relationships. Typically, in this phase we delayed decomposing the complex Object 1 and 2 as shown below.

SEA Online **College** —has—many **student**s and many **faculty**.
Each **student**— has — **ssn**, **name** and current **telephone number**
Each **faculty** — has — **unique name** and **office number**
Each **faculty** — teaches — **3 courses**
Students — send— **exams** to **faculty** .
Faculty — corrects — **exams** and computes **grades**.
Next he — sends— **grade** to the **gradebook**.
Gradebook— has— **grade** for each **course** and for each **student**.
Student — can see only their own **grades** in the **gradebook**.

In some cases the relationship can not be identified as in the example of the last sentence. In the next phase several processes take place such as accepting some compound objects, splitting other compound objects, identifying number of object instances in the concept, and attaching qualifiers to appropriate concepts. This is shown on the example of the transformation of the first tuple (first sentence).

SEA Online College —has—**Student** (many)
SEA Online College —has—**Faculty**(many).

SEA Online College —has one instance .
Student —has N instances
Faculty — has N instances

Similarly we process the second tuple (second sentence)

Student— has — **ssn (unique)**
Student— has — **name**
Student— has **telephone number** (current)

While attaching qualifiers to appropriate concepts, we can take them from the text or from the ontology as shown above. We qualified **ssn** by the qualifier **(unique)** taken from the ontology.

Faculty — has — **name (unique)**
Faculty — has — **office number**

Faculty — teaches —**Course (3)**
Course — has N instances

Student — sends— **exams** to **Faculty** .
Faculty — corrects — **exams** and computes **grades**.

" He" is replaced by the last tuple concept as shown below but other changes are similar:

Faculty — sends— **grade** to the **Gradebook**.

Gradebook— has— **grade** for **Course** and **Student**.
Gradebook— **has N instances**.

Student — can see only their own **grades** in the **Gradebook**.

The set of above tuples represents an initial knowledge database that can be used for generation of information system design. Before it is actually done some additional transformations will be necessary.

4. Query Module

In the second stage, the query processor is used to verify the consistency of knowledge contained in internal ontology and internally represented documents. The conflicts are resolved either automatically (if possible) or manually as shown in Fig. 4.

The queries can be simple, complex, definition, comparison, or summary. The simple queries would deal with properties, functions, and direct relationships between class types. The complex queries can involve indirect relationships. The definition queries would require identifying all properties, functions and direct relationships for a chosen class. The comparison queries would request a comparison of classes. The summary queries would create an entire document from the UML diagram.

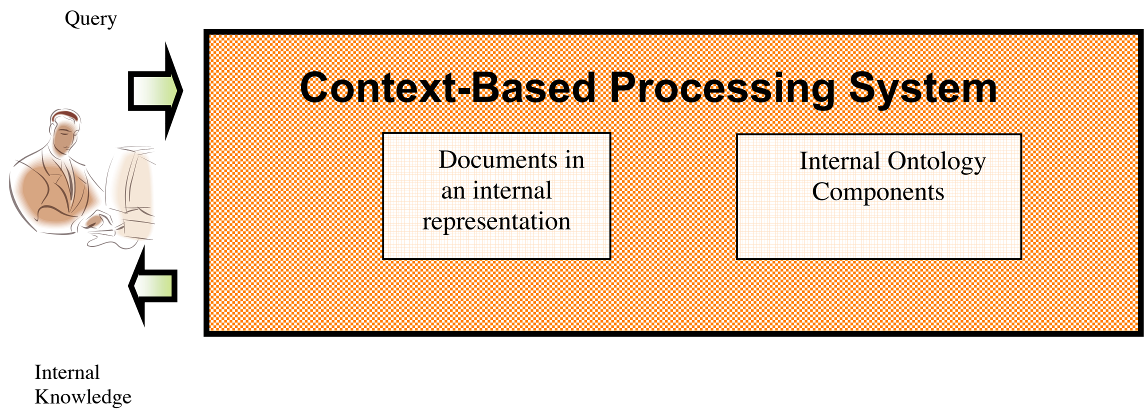

Query
Context-Based Processing System
Documents in an internal representation
Internal Ontology Components
Internal Knowledge

Figure 4: Query Module.

An example of a simple query could involve an aggregation relationship (e.g., "What are the components of SEA Online College?"). Other simple query could involve other types of relationships (e.g., "What receives grade from Faculty?").

An example of a complex query would be to identify the relationship between Student and Faculty. An example of a definition query would be to ask for a definition of SEA Online College. An example of a comparison query would be to ask: "What are similarities between Student and Faculty?" Additionally, we could query a UML diagram to create an entire document. The answer would be obtained by traversing the UML diagram from top to bottom, collecting all definitions.

5. UML Diagrams for Knowledge Modeling

The UML was created and used for software design. This kind of modeling is *object oriented*, meaning that whatever system is being modeled, its components become abstract objects that have some properties (also called *attributes*) and functions (also referred to as *responsibilities*). A *class* is a collection of these abstract objects. Generally, there are several types of UML diagrams that can be useful for knowledge modeling, including class diagrams and state diagrams. In this paper, we will concentrate on class diagrams.

Class diagrams contain classes and relationships. Classes can be described by their name, properties, and functions, and they are graphically represented as boxes. Lines or arrows are then drawn between classes to describe their relationships, the most common of which are aggregation, generalization, and named association. Typically, aggregation is treated as a special form of association [4, 20], but since aggregations play an important role in this paper, for convenience we will discuss aggregations separately from other associations, which are referred to here as *named associations*.

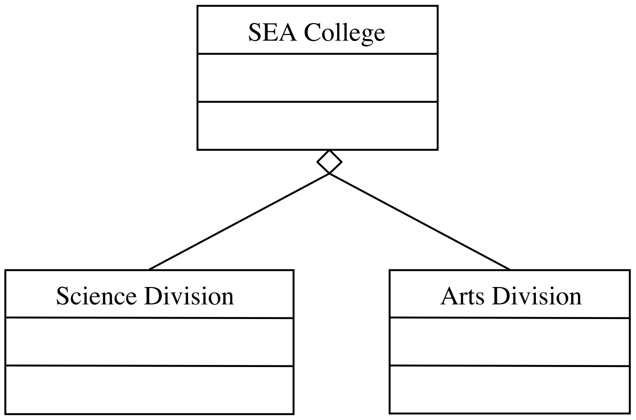


Figure 5: Knowledge Represented by a UML Class Diagram

Let us consider a UML diagram describing common knowledge about SEA College as shown in Fig. 5. The diagram simply states two important concepts (classes), SEA College and Division, and that a division is a part of SEA College. The diamond specifies an aggregation relationship. Similarly we can specify different relationships using different symbols as shown in Figure 6.

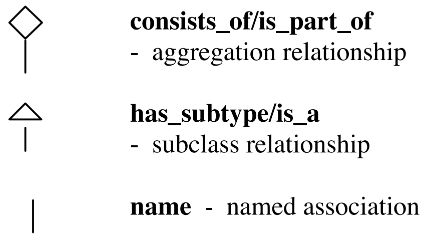


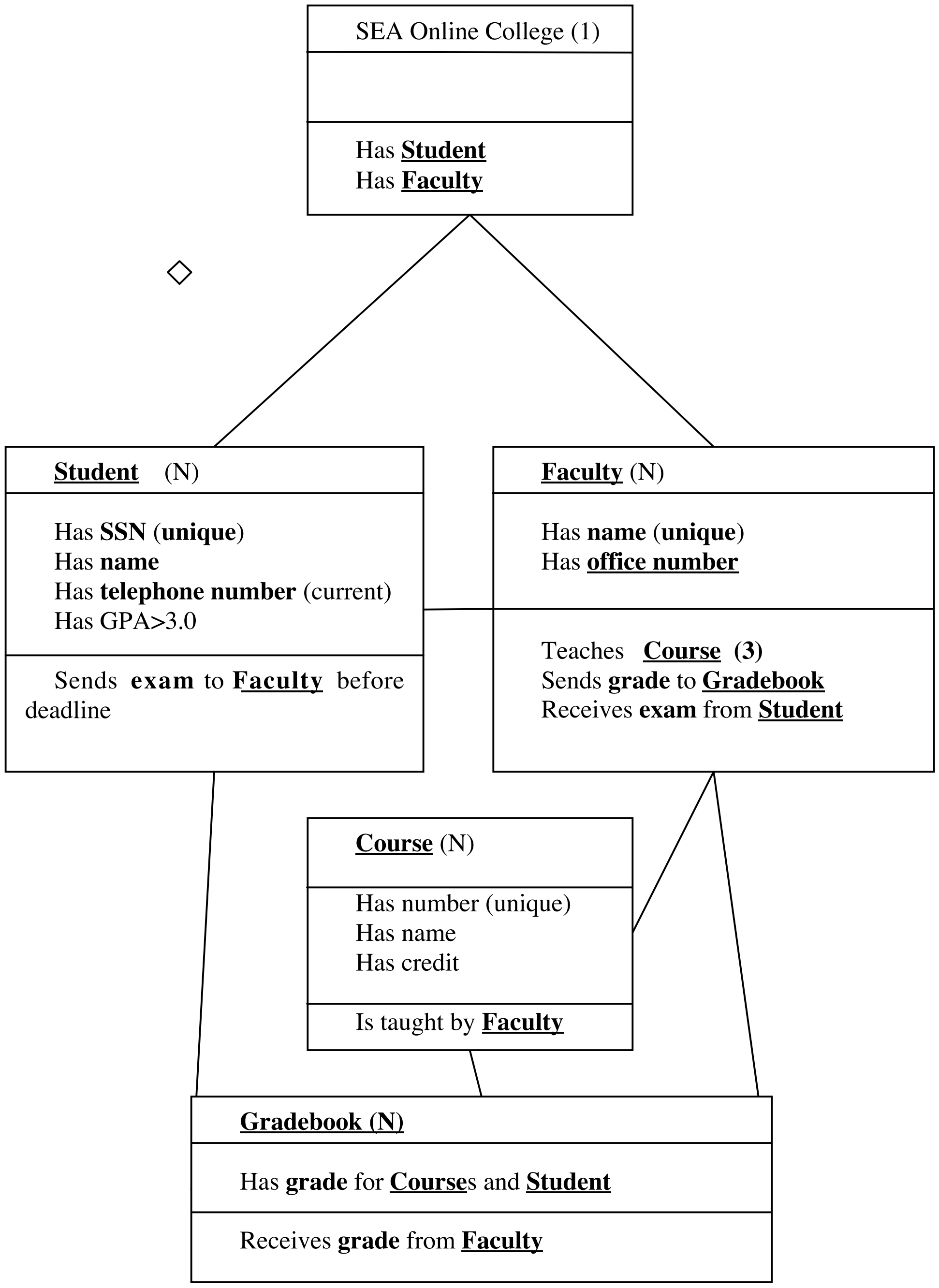
Figure 6: UML Relationships

In order for UML diagrams to capture knowledge from various subject areas, two problems need to be resolved. The first problem is defining a consistent translation methodology for transferring information from a natural language into a diagrammatic form. The second problem is to extend the UML class diagram to allow the student to capture all knowledge in such a form.

The UML class diagrams can be used to represent knowledge database containing well-structured knowledge facts. By well-structured knowledge facts, we mean tuples consisting of three components: first class name, relationship and second concept that can be either class name or a property. The class name can be expanded by a multiplicity constraint that can be a typical numerical constraint or more descriptive such as “many,” “always,” “often,” etc. The UML diagram corresponding to our sample set of tuples is shown in Fig. 7.

UML diagrams are very important to visualize knowledge database for quick visual inspection of a human expert. They can also assist in knowledge database integration and cleaning. The knowledge integration is mainly related to identifying common classes and refining nonstandard tuples. The knowledge cleaning is mainly related to finding similarity in functions/relationships and determining which are identical, which are refinements, and which are different.

6. Creating UML Diagrams for Knowledge Modeling



SEA Online College (1)
Has **Student**
Has **Faculty**
Student (N)
Has **SSN** (**unique**)
Has **name**
Has **telephone number** (current)
Has GPA>3.0
Sends **exam** to **Faculty** before deadline
Faculty (N)
Has **name** (**unique**)
Has **office number**
Teaches **Course** **(3)**
Sends **grade** to **Gradebook**
Receives **exam** from **Student**
Course (N)
Has number (unique)
Has name
Has credit
Is taught by **Faculty**
Gradebook (N)
Has **grade** for **Course**s and **Student**
Receives **grade** from **Faculty**

Figure 7. The Intermediate UML Diagram for a Sample Text

Let us discuss first the translation of nonstandard class descriptors such as the one listed below "**Gradebook (N)** Has **grade** for **Course**s and **Student**

The system can interactively guide the human expert to identify whether the relationship is n-ary or whether it is a composite of many relationships. If the n-ary relationship is selected, the system can ask about:

1. active and passive classes;
2. the role (function) of each active class.

Classes in general are classified into active and passive. Even though the relationship is n-ary, meaning that it involves all n classes, only active classes have the function associated with the relationship. The necessary cleaning phase in our case required identifying similarity of functions/relationships like “send,” “take,” etc.

7. Database Design and Constraints Generation

Based on UML diagram the database schema can be generated as shown in Fig. 8. For our example of text the following database schema is generated:

Student (<u>SSN</u>, sname, telephone)
Faculty (f<u>name</u>, office_number)
Course (number, cname, credit, fname)
<u>Gradebook (</u>grade, <u>cname, SSN)</u>

Additionally, database constrains can be also generated e.g.

<u>Faculty</u> Teaches <u>Course</u> (3)

Also, general constraints can be generated to guide design of other aspects of Information Processing system e.g.

<u>Student</u> Has telephone (**current**)

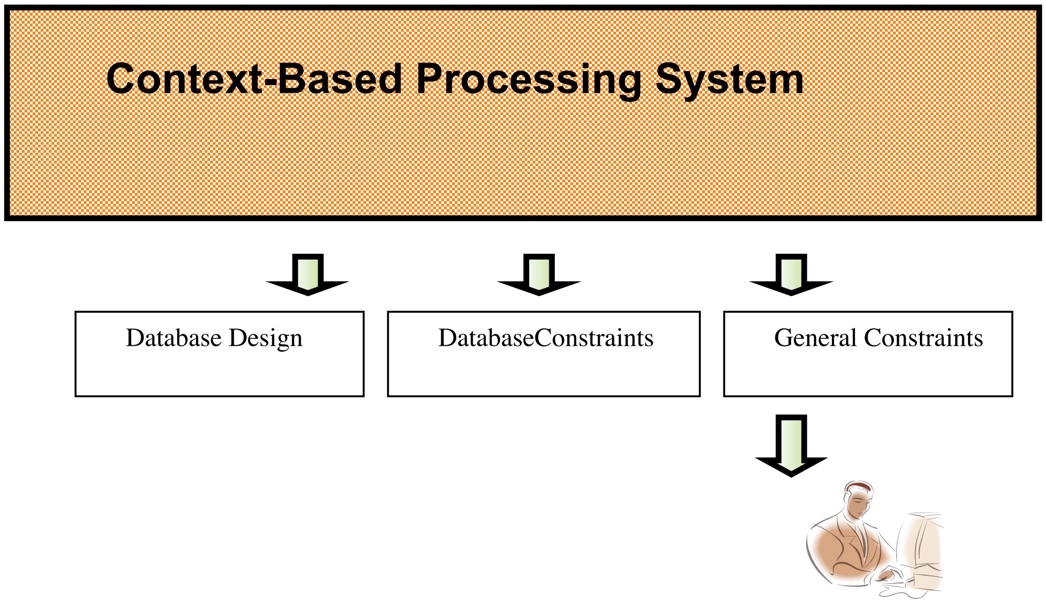


Figure 8: Database Design and General Constraints Genereation

The general constraint that the telephone number should be current can be easily overlooked in the design of an information system but by explicitly having it in the requirements list we can be forced to act upon it.

Another constraint: "Student Sends exam to Faculty **before deadline"** could trigger some design decision of information system to include this restriction explicitly. The same is applicable to the general constraint: "Faculty **Sends grade** to Gradebook"

8. Summary

Whenever humans process an individual document (text) they view it in a broad context of facts and rules acquired throughout our life. The same is applicable to the process of designing information systems. Our understanding of requirements document very much depends on the "scope" of our background knowledge. Information systems development involves processing a large number of documents containing a substantial amount of domain knowledge. Most of these documents are written in natural language and usually contain ambiguity, contradictions and redundancy. These documents can also be incomplete. The background knowledge can help resolve these problems. Additional important application of the background knowledge is that help in "fact mining" for facts that can be omitted during information systems development. Our solution is based on external ontology representing background knowledge. Guided by this ontology the text mining techniques are used to extract domain knowledge from provided documents. We use an extended Unified Modeling Language (UML) for graphical representation of knowledge modeling. Such knowledge includes not only "directly implementable" knowledge that can be converted into traditional database model, but also general constraints that should be in some phase taken into consideration while developing information systems.

References

1. Arnheim, R.: Visual Thinking. University of California Press, Berkeley, CA (1969)
2. Bemers-Lee T., Hendler, J., Lassila, 0.: The Semantic Web. Sci Am 284(5). (2001) 34-43
3. Brachman, R.: On the Epistemological Status of Semantic Networks. In: Findlee, N.V. (ed): Associative Networks: Representation and Use of Knowledge by Computer. Academic, New York (1979) 3-50
4. Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modeling Language User Guide. Addison Wesley, Reading, MA (1999)
5. Chein, M., Mugnier, M.L.: Conceptual Graphs: Fundamental Notions. Rev Intell Artif 6 (4). (1992) 365-406
6. Czejdo, B., Mappus, R., Messa, K.: The Impact of UML Diagrams on Knowledge Modeling, Discovery and Presentations. Journal of Information Technology Impact, Vol.3 (1). (2003) 25-38
7. Czejdo, B., Czejdo, J., Lehman, J., Messa, K.: Graphical Queries for XML Documents and Their Applications for Accessing Knowledge in UML Diagrams. Proceedings of

the First Symposium on Databases, Data Warehousing and Knowledge Discovery. Baden Baden, Germany (2003) 69 – 85

8. Czejdo, B., Czejdo, J., Eick, C., Messa, K., Vernace, M.: Rules in UML Class Diagrams. Opportunities of Change. Proceedings of the Third International Economic Congress. Sopot, Poland. (2003) 289-298
9. Czejdo, B., Sobaniec, C.: Using a Semantic Model and XML for Document Annotations. Lecture Notes in Computer Science. Springer-Verlag, Berlin Heidelberg New York (2000) 236-241
10. Delteil, A., Faron, C.: A Graph-Based Knowledge Representation Language. Proceedings of the 15th European Conference on Artificial Intelligence (ECAI), Lyon, France (2002)
11. Dori, D.: Object-Process Methodology -a holistic systems paradigm. Springer, Berlin Heidelberg New York (2002). www.ObjectProcess.org
12. Dori, D.: Why Significant Change in UML Is Unlikely. Commun ACM 45 (11). (2002) 82-85
13. Dori, D., Reinhartz-Berger, I., Sturm, A.: OPCAT—A Bimodal CASE Tool for Object-Process Based System Development. Proceedings of the IEEE/ACM 5th International Conference on Enterprise Information Systems (ICEIS 2003), Angers, France (2003) 286-291. www.ObjectProcess.org
14. Lehrnan, F. (ed): Semantic Networks in Artificial Intelligence. Pergamon, Oxford, UK (1999)
15. Mayer, R.E.: Multimedia Learning. Cambridge University Press, New York (2002)
16. McTear, M.F. (ed): Understanding Cognitive Science. Ellis Horwood, Chichester, UK (1998)
17. Novak, J.D.: A Theory of Education. Cornell University Press, Ithaca, NY (1977)
18. Novak, J.D., Gowin, D.B.: Learning How to Learn. Cambridge University Press, New York (1984)
19. OMG UML1.4. Object Management Group Unified Modeling Language v.l.4 (2001)
20. Rumbaugh, J., Jacobson, I., Booch, G.: The Unified Modeling Language Reference Manual. Addison-Wesley, Reading, MA (1999)
21. Rumbaugh, J.: Objects in the Constitution: Enterprise Modeling. Journal of Object Oriented Programming. (1993)
22. Schulte, R., Biguenet, J.: Theories of Translation. The University of Chicago Press, Chicago (1992)
23. Smith, M.K., McGuinness, D., Volz, R., Welty, C.: Web Ontology Language (OWL) Guide Version 1.0. W3CWorking Draft (2002)
24. Sowa, J.F.: Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, Reading, MA (1984)

25. Sowa, I.F.: Knowledge Representation: Logical, Philosophical, and Computational Foundations. Brooks Cole, Pacific Grove, CA (1999)

26. Walpole, B.: Pocket Book of The Human Body. Wanderer, New York (1987)

A Distributed System for Clustered Visualization: A New Paradigm for Information Modeling

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

ASR International Corporation[3]
Hauppauge, NY, USA

Abstract

Large memory and storage capacities and ever more powerful processors have enabled us to accumulate data at a pace faster than ever. Huge databases collect data ranging from applications as critical as air traffic flow, to as mundane as user search patterns on the web. With more and more data, extracting useful information from it is ever so harder. We propose a new approach to collate and represent data as information in a meaningful way, through graphical, auditory, and textual representations. We further propose a self-correcting correlation matrix that can isolate trends and patterns in the existing data, and make future projections. Such projections can assist managers in making business decisions, and in time can lead to savings in both time and money.

In today's digital world, globalization has pushed the business paradigm to a distributed workforce that is mobile and highly diverse. For managers to be able to make real-time decisions that can have ramifications at different physical locations at different points of time, the transformation of data to information to knowledge to wisdom has to be real-time as well. We will demonstrate a web-based secure system which **collect**s data uploaded by remote employees (documents, spreadsheets, in standard formats), which is **process**ed and analyzed at the central servers, for managers to access and **consume** on-demand.

Such a **collect-process-consume** model effectively shields the complexity of the data-crunching and analysis from the end-users. Managers can either make a decision based on the bird's eye-view provided to them by the system, or further drill-down into the data using graphs and other visual aids for more details. Managers can choose to receive periodic analysis and suggestions regarding the performance of the network of distributed employees by email, fax, voice, and/or SMS (text messages). Over time, the data accumulates into a knowledge base, which the managers can refer to for institutionalized wisdom as well.

Keywords

Distributed Systems, Database Systems, Visualization, Analysis, Correlation, Prediction

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

Introduction

As memory / storage capacities and processing powers grow exponentially, collecting and storing data are no longer the bottlenecks they once were. In today's digital age, we are able to capture data from myriad sources and in different formats, be as pictures, videos, documents or data-sheets (schedules, working hours, and system performance indicators, to name a few). High-capacity networks have enabled data to be stored and retrieved from centralized servers in an *on-demand* fashion, with relatively negligible transmission times.

However, as more and more data are accumulated, the issue of extracting meaningful information from them and to make business decisions in a *timely manner* becomes increasingly more critical. In addition, in today's distributed workforce environments, the data might be generated at different physical locations, and at different organizational levels. The goal is to gather and process this data *live*, to give managers and other data consumers the required information as fast as possible.

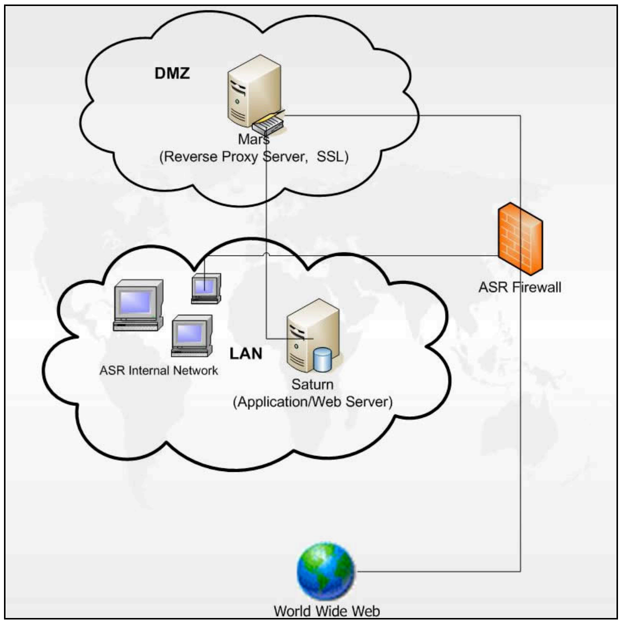


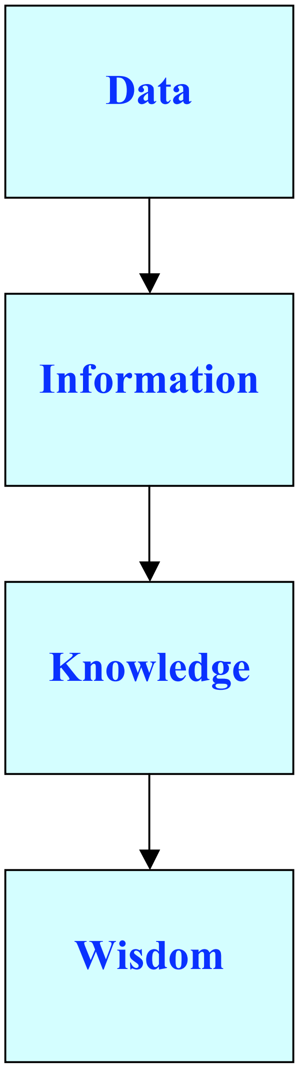
Figure 1: Network Layout

To this end, we have developed **P3M**, our proprietary **P**rocess/**P**erformance/**P**roject **M**anagement system. Running on a central application server (Figure 1: Network Layout) exposing a web interface through a reverse proxy server, the system allows remote users to securely log in and upload documents and data. These data are extracted, pre-processed and validated at the server and the extracted information stored in a database. Multiple levels of access let users to access information (raw-data, graphs, analyses, audio, notifications, etc.) at different levels of detail. To facilitate this process, we have a slew of in-house tools, within the auspices of what we call the **ASR Technology Platform** (**ATP**).

ASR Technology Platform

Data is everywhere. We collect data ranging from the mundane (web usage patterns) to the critical (traffic flows), but, in absence of any further processing, these huge collections of data are nothing but glorified bits and bytes stored in some hard-drive. Therefore the critical business need is for a system that:

- Collects **Data** with minimal or no impact on the overall functioning of the system it is collecting data on
- Efficiently extract **Information** from this data
- Somehow transfer this information to the authorized personnel in a *timely* manner to be used as a **Knowledge** base, and
- Have an efficient correlation and prediction architecture, that
  - Correlates collected information over a period of time to find patterns,
  - Use these identified patterns to do short and long-term predictions,
  - Provide through these accumulated knowledge and correlation/prediction tools, a semblance of an institutionalized **Wisdom**.



- Data is collected at various sites across the country. The data might be in the form of
  - Log of user activities as they log on to the central server from the computers at their sites,
  - Structured data regarding site performance, uploaded as documents or spreadsheets,
  - Online forms containing questions, filled up by the users on a periodic basis.

- The uploaded data/documents are processed at the server.
  - **Identify**: the data is first checked to see if it is what it is supposed to be,
  - **Validate**: Check the data for internal consistencies; e.g. an employee must have data on hours worked if he/she has marked *Present* for that day,
  - **Extract**: Once validated, the files are parsed and data is extracted,
  - **Store**: The extracted data is stored on a database as structured information.

- Once stored, this information is presented to the authorized users through the following specialized modules.
  - **Visualization**: Present the information collected over a selected period of time, in charts and graphs.
  - **Business Rules**: Customized rules that govern what information is to be visualized, and how.
  - **Audio**: Audio summaries are generated that can be listened to over a mobile device.
  - **SMS**: Information on critical parameters can be text-messaged to the managers on a need-to-know basis.

In today's distributed workplace, it is critically essential that data is shared, distributed and analyzed in a seamless fashion. However, even though it is easier to share data (through email or intranets), extracting information from data and using that information as knowledge are often the bottlenecks that stifle the efficient operation of a workplace. For example, a manager might like to know how often his employees (who are spread across a vast region) are logging on to a system, to perform certain day-to-day critical tasks. This information might then be used to dynamically allocate more (or less) manpower at the various sites. Unless this information is updated in a real-time fashion, the manager's response will always lag behind the actual needs of the sites, and thus productivity will suffer.

In addition, this collected data has to be presented to the manager in a form that is pre-processed and easily digestible. Thus the system should not only collect the data with little or no effort on part of the employees (collection of data should not be detrimental to the overall efficiency), but also be able to process and present the data as information in near-real-time. We achieve this though our proprietary technology platform (Figure 2: ASR Technology Platform, which is modeled on a Client-Server framework.

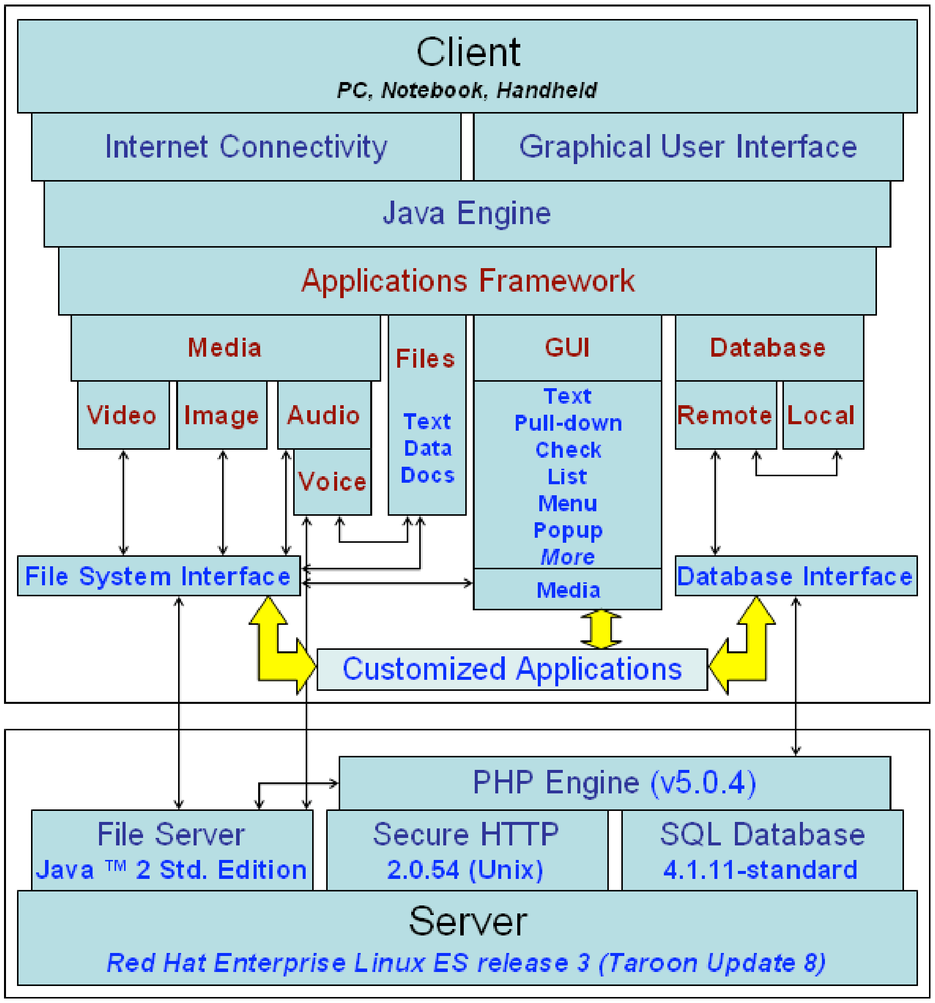
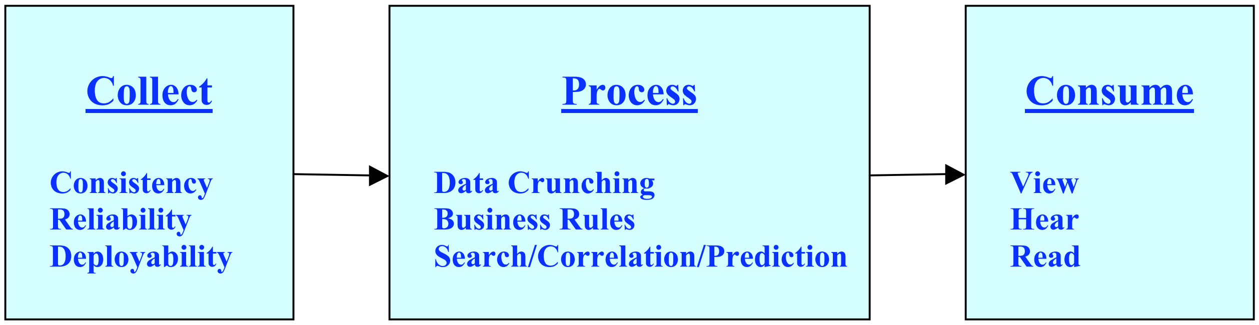


Figure 2: ASR Technology Platform

The platform consists of a central server which exposes a secure (128 bit AES) web interface for remote users to log in using their web browsers. Once in the system, they have access to (depending upon their access levels, which is set by the administrator) different modules for various tasks. Their online activities are recorded on a server-side database. The remote users also upload documents/data-sheets that are processed and parsed by Java-driven services on the server, and the extracted data is added to an online database. This data is used to visualize and present information to the higher users (managers, administrators) who can use this processed information (presented in a visual or auditory format) to drive their decisions.

Running on top of this ATP architecture is our P3M framework that consists of the web and Java tools that enable the users to interact with the system in this coherent fashion. The P3M framework is based on a **collect-process-consume** (**CPC**) model, where users are producers of data and consumers of information, with the central server acting as the intermediary that processes the uploaded data for easy consumption at different levels throughout the hierarchy. This CPC model is at the heart of the P3M framework.

CPC Model



- **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

Results

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 visualizations can be viewed either using a web browser (from a computer or web-enabled phone). For the mobile application, a java applet can be downloaded which will facilitate asking such questions. In this section, we display some of the possible visualizations, and discuss the inference/trends that manifest themselves. To ensure privacy, we have scrambled some of the data/axis labels in the plots. Doing so however does not in anyway dilute the thrust of the results.

First we look at data regarding user logins to the system, during January-April 2007. The information gleaned from such data is used by the managers to decide if the online resources are being judiciously used at different sites.

- The plot (Figure 3: Site by Site system usage.) gives the breakup of system usage at different sites (for January – April 2007).

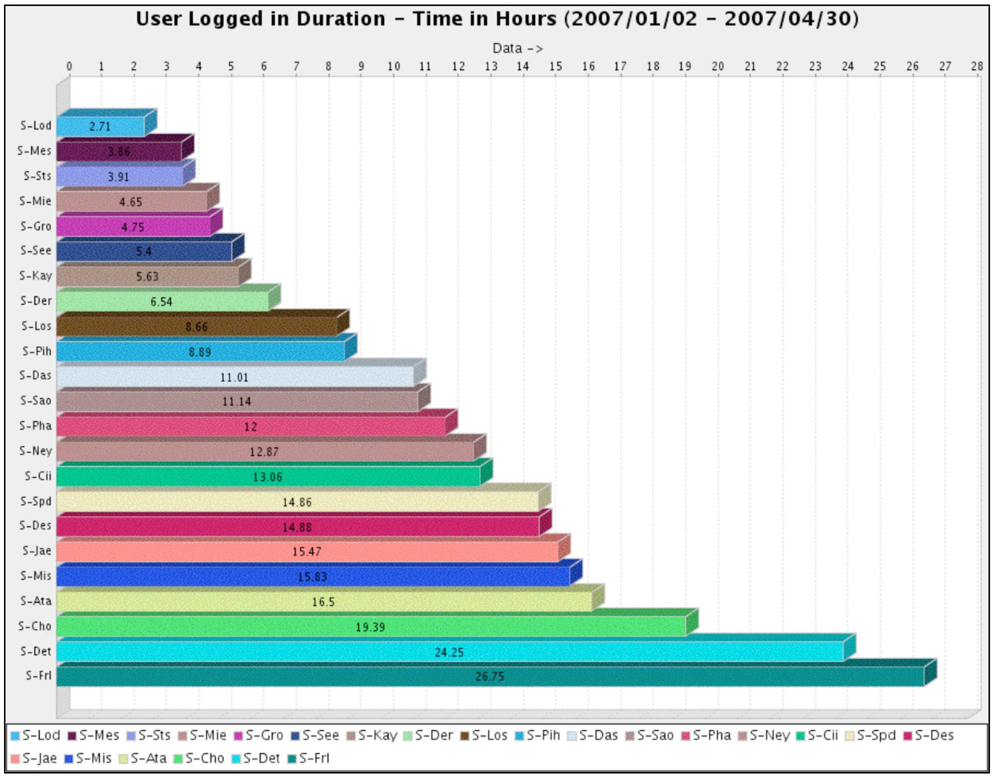


Figure 3: Site by Site system usage.

- The data is visualized to give *month-wise* information (Figure 4: Month-wise system usage.). The usage was higher in January, as we introduced some new updates which required the users to spend more time on the system. This information can be to analyze how much time is utilized to learning new system features, as opposed to basic usage.

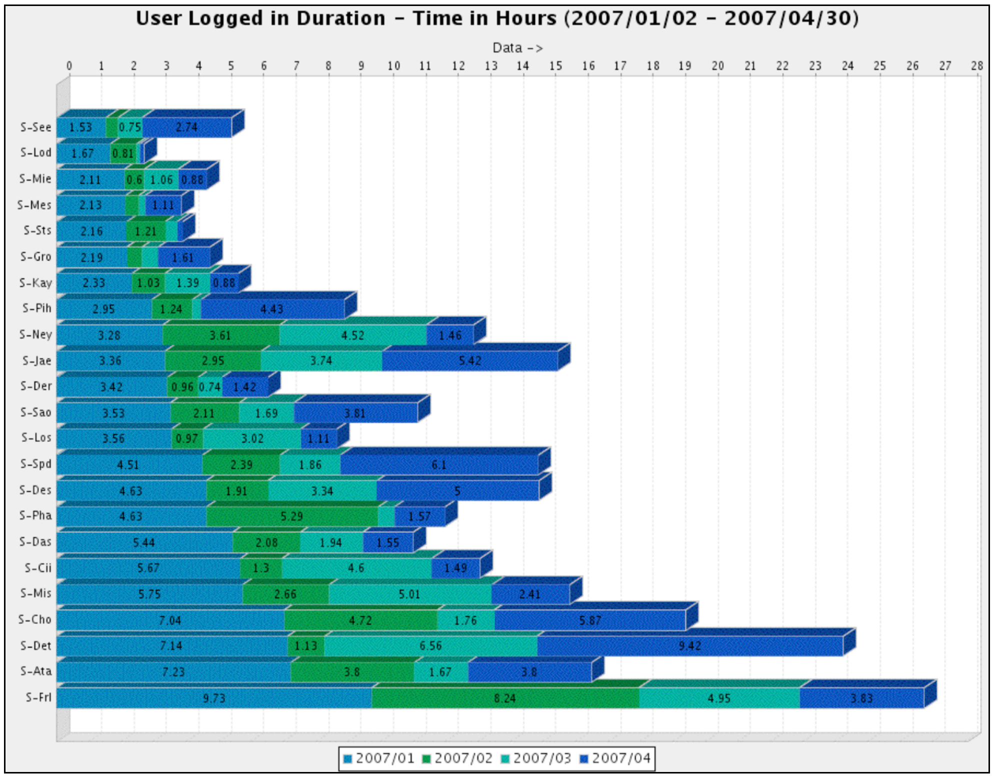


Figure 4: Month-wise system usage.

- More specifically, in the following plots (Figure 5: Month/Week-wise system usage.) we can see a greater system usage during the month of January, as opposed to the later months. Another spike was in the month of April, when another update was introduced.

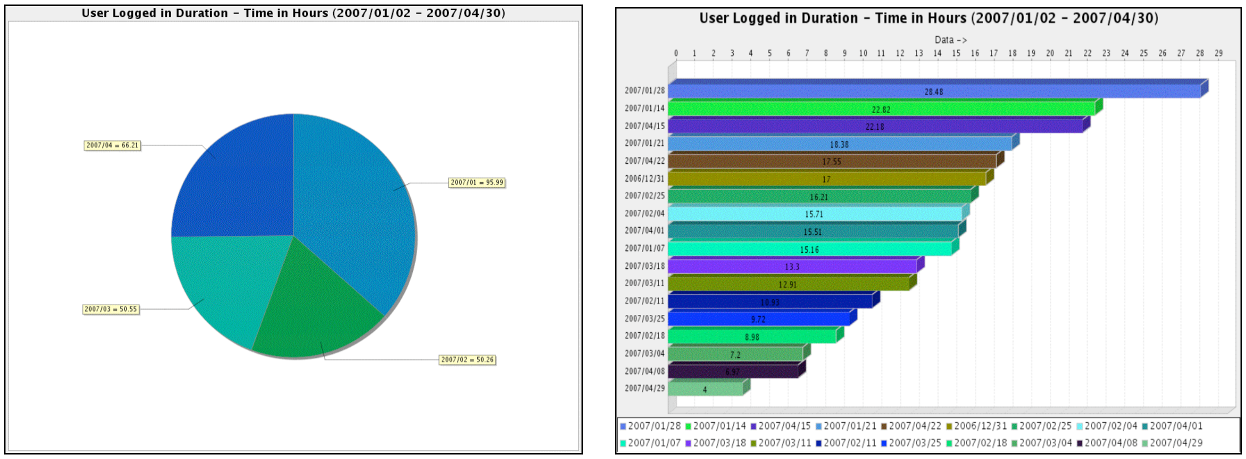


Figure 5: Month/Week-wise system usage.

- From time to time, there is a need to see if usages at different sites rise and fall together. This is important, as this suggests that perhaps the sites are facing similar problems. We have developed a clustering tool that computes correlations and aggregates sites according to their similar usages (Figure 6: Correlated site usages.).

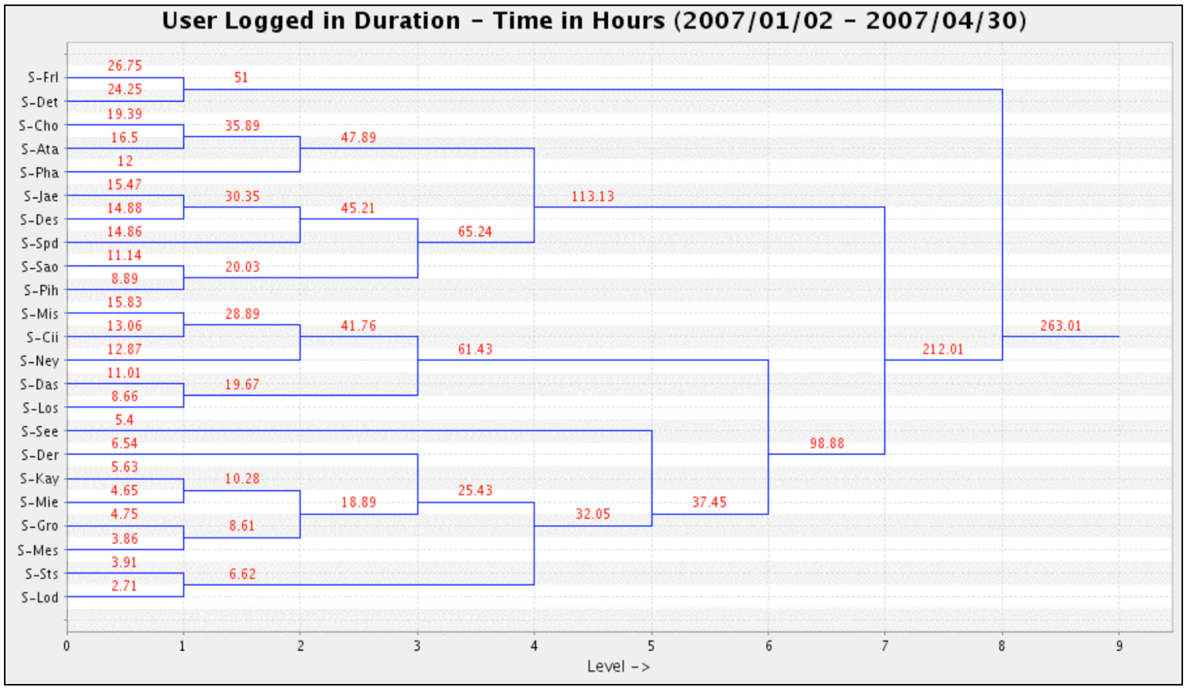


Figure 6: Correlated site usages.

- In addition, cross-correlations (Figure 7: Correlation between two data parameters) between different data can suggest if the sites are operating at optimal efficiency. For example, the total work-time spent by workers at a site should in general proportional to the amount of work-load.

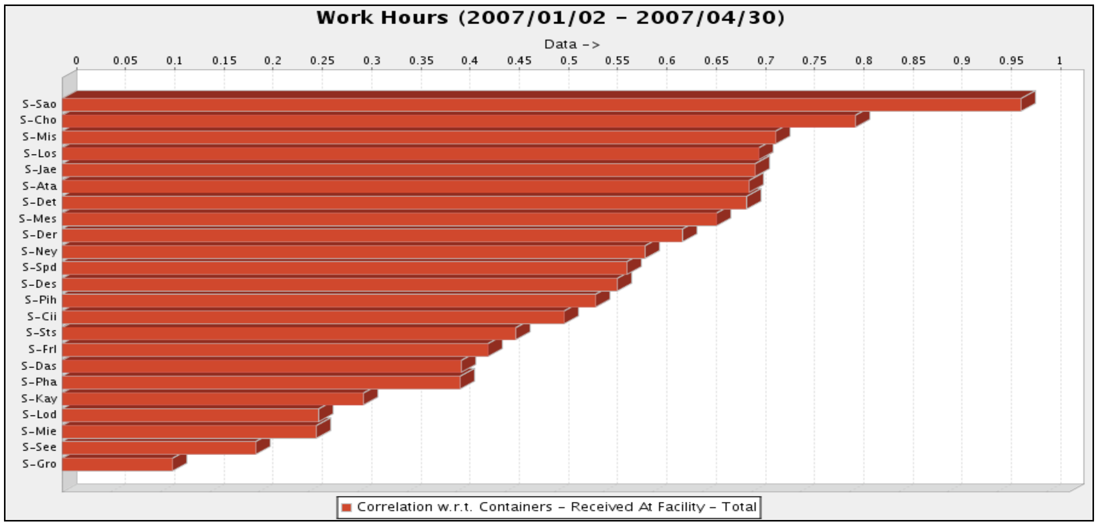


Figure 7: Correlation between two data parameters.

Conclusion

As we accumulate more and more data at an ever increasing rate, the capability to collect data is outpacing the capability to process and analyze it. Raw data is often useless, unless information can be extracted within a set period of time. Moreover, in a distributed workplace, data often has to be gathered under certain time-constraints as well, since it is of little use if one has exceptional analysis tools if the raw data could not be accumulated in time.

Our distributed web-based ATP framework achieves both of the above through a **CPC** (Collect-Process-Consume) model. Data is **collect**ed at remote sites through either user uploads, or user activities on a secure web-site. The backend server validates and **process**es this data into information that is accumulated in an on-server database. This information is **consume**d in the form of document summaries, audio readouts, text messages and/or graphical visualizations by the managers, who either connect to the server using a secure login, or have this information emailed/messaged to them.

Thus the data is processed and converted into manageable information that the managers can use to help them manage the workforce. Additional server-side tools such as for analyzing correlations and construct trends also help the managers to do short and long-term predictions towards increasing the overall efficiency of the system. Manager-specific specialized business rules are custom-made, to show managers information on data that are most important to them.

Such a system has two major benefits. First, the manager is shielded from the volumes of data that are generated and stored each day. Instead of data, the manager need only deal with the compact set of information generated by the system. Specialized tools along with business rules ensure that the manager get a snapshot of the enterprise whenever and wherever needed. Second, the system is scalable as all the process and analysis are done at the server-side (whose resources can be scaled up as needed), and web-based, thus letting end-users connect with devices as diverse as laptops to PDAs to mobile phones.

Future Directions

In future, the system can be further enhanced to introduce AI-based modules that can analyze data in a much more robust way to find patterns, and identify future trends. Such an approach can have immense benefits in data-intensive systems such as the stock-market, long-term weather-forecasting, and air-traffic control.

In addition, the flow of information can be made bidirectional, which will let employees at remote sites, who right now are just generating and uploading data, also get a feel of their performance as relative to the overall operation of the system. If tied with a performance-based rewards scheme, this can create a feedback-based model that can maximize the efficiency of the enterprise over a reasonable period of time.

Theory of K-Representations as a Framework for Constructing a Semantic Networking Language 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, 105187 Moscow, Russia and IIAS
E-mail: vdrfom@aha.ru and vfomichov@hse.ru
Fax: +7-495-771-3238

Abstract

Since the middle of the 1990s, UNO has been funding a large-scale project aimed at the design of a family of natural language processing systems transforming the sentences in various natural languages into the expressions of a new language-intermediary called the Universal Networking Language (UNL) and also transforming the UNL-expressions into sentences in various natural languages. In this paper it is shown that the expressive possibilities of UNL are rather restricted; first of all, from the standpoint of representing the meanings of discourses and representing knowledge about the world. That is why it is concluded that the real content of the mentioned large-scale UNO project is the creation of an initial version of a Universal Semantic Networking Language (USNL) and of the transformers from natural language (NL) to this initial version and from this version to NL.

This paper proposes a new way for developing a USNL. This way is to use the definition of the class of SK-languages (standard knowledge languages) given by the theory of K-representations (it was developed by the author and represented in numerous papers in English and Russian) as a model of a Semantic Networking Language of a new generation in comparison with UNL. The examples of building semantic representations of the natural language texts and of representing knowledge pertaining to medicine, biology, and business are considered.

Keywords

UNL; universal semantic networking language; natural language processing; semantic representation; theory of K-representations; SK-languages

Introduction

During last decade, one has been able to observe the stormy progress of the Internet not only in developed countries but also in many tens of developing countries. As a consequence, the following fundamental problem has emerged: how to eliminate the language barrier between the end users of the Internet in different countries. For solving this problem, H. Uchida, M. Zhu, and T. Della Senta (Uchida et al. 1999) proposed in the middle of the 1990s a new language-intermediary, using the words of English language for designating informational units and several special symbols. This language, called the Universal Networking Language (UNL), is based on the idea of representing the meanings of separate sentences by means of binary relations. The *second motive* for the elaboration of UNL was an attempt to create the language means allowing for representing in one format the various pieces of knowledge accumulated by the mankind and, as a consequence, to create objective preconditions for sharing these pieces of knowledge by various computer systems throughout the world.

Since 1996, UNO has been funding a large-scale project aimed at the design of a family of natural language processing systems (NLPSs) transforming the sentences in various natural languages into the expressions of UNL and also transforming the UNL-expressions into sentences in various natural languages. During several years the coordinator of this project was the UNO Institute for Advanced Studies by the Tokyo University. Since the beginning of the 2000s, the studies in this direction have been coordinated by the Universal Networking Digital Language Foundation. At the moment, under the framework of this project, the NLPSs for six official UNO languages are being elaborated (English, Arabic, Spanish, Chinese, Russian, and French), and also for 9 other languages, including Japanese, Italian, and German. Since the beginning of the 2000s, the studies in this direction have been coordinated by the Universal Networking Digital Language Foundation.

The initially scheduled duration of the UNL project started in 1996 is ten years. That is why it is just the time to analyze the achieved results and to take the right decisions concerning the further studies in this direction. Continuing the line of (Fomichov 2004) with respect to the online monographs (UNL 2005, 2005), this paper shows that the expressive possibilities of UNL are rather restricted. That is why it is proposed to interpret the language UNL (despite of the linguistic meaning of its title) as a semantic networking language of the first generation. The outlines of the expressive possibilities of the class of SK-languages, or standard knowledge languages, defined in (Fomichov 1996), are given in (Fomichov 2002, 2005b). The definition of the class of SK-languages is one of the basic constituents of the theory of K-representations elaborated by the author of this paper (Fomichov 2005a). As a result of comparing the expressive possibilities of SK-languages and UNL, it is proposed to consider the definition of the class of SK-languages as a model of a semantic networking language of the next generation.

UNL as an Initial Version of a Semantic Networking Language

UNL represents sentences in the form of formal expressions, without ambiguity, destined not for humans to read, but for computers. The purpose of introducing UNL in communication networks is to achieve accurate exchange of information between different languages. Information has to be readable and understandable by the users.

The initial version of UNL (1997 – 2004) was oriented at representing the contents of only separate sentences but not of discourses. The examples of the basic constructions used in this version of UNL are the expressions

tower (icl > building), murano (icl > thing-out-of-glass, aoj >colour),

build (icl > do), obj (build (icl > do), tower (icl > building)) , where the strings *icl , aoj, obj* are interpreted as the designations of the binary relations “A concretization of a concept”, “An attribute of a thing”, “The object of an action”. The UNL specifications published in the year 2005 (UNL 2005) introduced a manner to build the designations of compound concepts as the so called scopes (this notion is analyzed in detail below). The step done in the year 2006 consists in adding the means allowing for representing the meaning of the idioms (UNL 2006).

The analysis shows that in fact the expressive possibilities of UNL are very restricted. First of all, the language UNL is oriented at representing the contents of only separate sentences but not arbitrary discourses. Even the UNL specifications published in 2006 don’t contain a theory of representing the meanings of discourses. Besides, UNL is inconvenient for representing, in particular, the meanings of sentences with complicated goals (being parts of advices, commands, wants, etc.), designations of sets, the word “notion”, homogeneous members of sentence. Let’s consider, for instance, the definition “A flock is a large number of birds or mammals (e.g. sheep or goats), usually gathered together for a definite purpose, such as feeding, migration, or defence”. An attempt to represent the meaning of this definition in the language UNL, i.e. with the help of only the designations of binary relations, would lead to a complete destruction of a connection between the structure of the considered definition and the structure of its UNL-representation.

The possibilities of using the language UNL for representing knowledge about the world are very restricted too. Thus, the expressive possibilities of UNL not completely but only partially correspond to its title “a universal networking language”. That is why it seems to be reasonable to interpret the language UNL as not final but only *initial* version of a semantic networking language. The demands of formally representing the meanings of complicated discourses (for example, pertaining to medicine, science, technology, business, ecology, low) and the demands of automatic conceptual processing of semantic representations of such texts with respect to a knowledge base are to lead in the nearest future to the elaboration of a semantic networking language of a new generation. Hence it is reasonable to look for another, more powerful formal approaches to describing meanings of natural language texts with the aim to find (if possible) a model for constructing a universal or widely applicable semantic networking language.

Shortly About the Theory of K-representations

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 1992 – 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 2002, 2005b). The examples considered in this paper use the class of restricted SK-languages completely defined in (Fomichov 1996).

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) + Preposition + Noun”, “Verbal form+ Noun”, “Noun1 + Preposition + Noun2”, “Noun1+ Noun2”, “Number designation + Noun”, “Attribute+Noun”, “Interrogative word + Verb”.

The *third basic constituent* of the theory of K-representations is a complicated, strongly structured algorithm carrying out semantic-syntactic analysis of texts from some practically interesting sublanguages of NL. This algorithm is based on the elaborated formal model of a LDB. Additional information about this algorithm can be found in the final section of this paper.

The Definition of the Class of SK-languages as a Model of a Semantic Networking Language of the Next Generation

The analysis shows that it is not difficult to approximate the basic expressive mechanisms of UNL by means of SK-languages, because one of the rules used in the definition of the class of SK-languages is destined for constructing formulas with the names of n-ary relationships and the other rule allows for building compound designations of notions.

Example. Let’s consider the UNL-expression *to(train(icl > thing), London(icl > city))* ; it denotes a train for London and is taken from (Uchida et al. 1999). This expression can be approximated by the K-string *E1* of the form

*Destination (certain train * (Concr, thing), certain city * (Name, ‘London’))*

or by the K-string *E2* of the form

*certain train * (Concr, thing)(Destination, certain city * (Name, ‘London’)).*

In a similar way it is possible to construct the K-strings

*tower * (Concr, building), murano * (Concr, thing * (Material, glass)),*
*construction * (Concr, action), Object (certain construction * (Concr, action),*
*certain tower * (Concr, building))* .

The achieved level of the studies on elaborating the language UNL and using UNL for representing the meanings of regular, non-idiomatic NL-texts is reflected in the monograph (UNL 2005), having been available online since June 2005. It should be noted that this monograph provides no analysis of the conceptual structure of discourses and, as a consequence, gives no recommendations about the construction of semantic representations of discourses. The progress achieved in the year 2006 (UNL 2006) concerns the representation of the meanings of idiomatic expressions. That is why the analysis carried out below is based on the publication (UNL 2005).

One of the most important steps on the way of expanding the expressive possibilities of the language UNL done in the work (UNL 2005) in comparison with the previous publications of the authors of this language consists in introducing the notion of a compound concept, or *scope*.

A scope is defined in (UNL 2005, 2006) as a finite set of binary relations grouped for expressing a compound concept. The analysis of the examples illustrating this definition shows that the authors mean distinguishing a set of the expressions of the form $R(c, d)$, where R is the name of a binary relation, c and d are the designations of the attributes of the relation R. The instrument of distinguishing such a set of the formulas is a mark v, inserted into each selected formula immediately after the name of the relation. Thus, a scope is a set of the expressions of the form

$$R_1 : v\,(c_1 , d_1),\ \ldots,\ R_n : v\,(c_n , d_n)\ ,\ \text{where}\ n > 1.$$

For example, in (UNL 2005, 2006) the sentence Sent1 = *[Women who wear big hats in movie theaters] should be asked [to leave]* is considered. The part of the sentence within square brackets is interpreted as the expression of what should be asked. The authors construct the following scope reflecting the meaning of two fragments within square brackets:

agt:01(wear(aoj>thing,obj>hat), woman(icl>person).@pl)
obj:01(wear(aoj>thing,obj>hat), hat(icl>wear))
aoj:01(big(aoj>thing), hat(icl>wear))
plc:01(wear(aoj>thing,obj>hat), theater(icl>facilities))
mod:01(theater(icl>facilities), movie(icl>art))
agt:01(leave(agt>thing,obj>place).@entry, woman(icl>person).@pl)

The analysis shows that the way of designating compound concepts proposed in (UNL 2005, 2006) has the following drawbacks.

1. The consideration of two fragments in square brackets as a whole and the construction of the corresponding scope contradict to the language intuition and to the traditions of theoretical and computational linguistics. The sentence Sent1 describes a situation of the type "asking". The following thematic roles (or conceptual cases) are realized in each situation e_k of such type: *Agent* (a relation between the situation e_k and an intelligent system who is asking); *Addressee* (a relation between e_k and an intelligent system or a finite set of intelligent systems who are being asked to carry out an action); *Goal* (a relation between e_k and an action to be performed in order to achieve certain goal). Taking this into account, it is unnatural to build a scope corresponding to the union of the fragments [*Women who wear big hats in movie theaters*] and [*to leave*]. Both our language intuition and the traditions of

theoretical and computational linguistics show the expedience of constructing a scope for the fragment [*Women who wear big hats in movie theaters*].

2. An essential drawback of the notion of a scope is that the form of a scope doesn't allow for distinguishing a scope designating certain set of entities from a scope designating the content of a complicated phrase or a discourse. Meanwhile, the discourses very often contain the expressions referring to the meaning of preceding or following phrases or larger parts of the text. That is why it is necessary to elaborate the means of expressing such references in semantic representations of discourses.
3. The next drawback of the notion of a scope is as follows. There are possible the situations when the set of the attributes of the relations belonging to a scope contains two or more entities qualified by the same concept (e.g., three ships). However, the monographs (UNL 2005, 2006) don't contain any analysis of such situations and give no recommendations about how to designate various entities qualified by the same concept and belonging to one scope.

The theory of K-representations provides such means of constructing the designations of compound concepts which allow for escaping the mentioned drawbacks of the notion of a scope in the specifications of the language UNL.

Example. In order to describe the set of all women wearing big hats at a certain moment in a certain cinema, it is possible to use the following K-string, i.e. an expression of a SK-language:

*certain set * (Quality-composition, woman)(Set-mark, S1)(Set-description, Properties-of-elements(arbitrary woman * (Element, S1) : y1, Situation(e1, dress-wearing * (Agent1, y1)(Object1, certain hat * (Size, big) : x1)(Place, certain cinema : x2)(Time, t1)))).*

The theory of K-representations possesses many additional important advantages concerning the construction of a semantic networking language of a new generation in comparison with UNL. Let's illustrate a number of such advantages.

Example. Let T1 = "A flock is a large number of birds or mammals (e.g. sheep or goats), usually gathered together for a definite purpose, such as feeding, migration, or defence". T1 may have the K-representation *Expr1* of the form

*Definition1 (flock, dynamic-group * (Qualitative-composition, (bird ∨ mammal * (Examples, (sheep ∧ goal)))), S1, (Estimation1(Quantity(S1), high) ∧ Goal-of-forming (S1, certain purpose * (Examples, (feeding ∨ migration ∨ defence))))).*

The analysis of this formula enables us to conclude that it is convenient to use for constructing semantic representations (SRs) of NL-texts: (1) the designation of a 5-ary relationship *Definition1*, (2) compound designations of concepts (in this example the expressions *mammal * (Examples, (sheep ∧ goal))* and *dynamic-group * (Qualitative-composition, (bird ∨ mammal * (Examples, (sheep ∧ goal))))* were used), (3) the names of functions with the arguments and/or values being sets (in the example, the name of an unary function *Quantity* was used, its value is the quantity of elements in the set being an argument of this function), (4) compound designations of intentions, goals (in this example it is the expression *certain purpose * (Examples, (feeding ∨ migration ∨ defence))* .

The structure of the constructed K-representation *Expr1* to a considerable extent reflects the structure of the definition T1. Meanwhile, any attempt to represent the content of this definition in the language UNL, i.e. with the help of only binary relationships, would destroy any similarity between the structure of T1 and the structure of its UNL-representation.

Example. Let T2 = "All granulocytes are polymorphonuclear; that is, they have multilobed nuclei". Then T2 may have the following KR *Expr2:*

(Property(arbitrary granulocyte : x1, polymorphonuclear) : P1 ∧
*Explanation(P1, If-then (Situation (e1, possessing1 * (Subject1, x1)(Object1,*
certain nucleus : x2)), Property(x2, multilobed)))) .

Here *P1* is the variable marking the meaning of the first phrase of T2; the strings *Subjectt1, Object1* designate thematic roles (or conceptual cases).

The key role in the construction of the K-representation *Expr2* was played by the rule enabling to introduce the mark *x1* for designating an arbitrary granulocyte, the mark *x2* for designating the nucleus of the cell *x1*, and the mark *P1* for designating semantic representation (SR) of the first sentence from the discourse _2. The mark (variable) *P1* enables to explicate in the structure of SR of T2 the reference to the meaning of the first sentence; this reference is given by the word combination “that is”.

The language UNL doesn’t provide the means for representing the meanings of sentences and larger fragments of discourses. Meanwhile, the last example considers one of the shortest discourses of the kind. The textbooks in various fields of knowledge contain a lot of much more complicated discourses with the references to the meanings of sentences and larger fragments of discourses.

Example (The possibility of constructing the compound designations of goals). Let T3 = “The owner of an insurance police calls the firm “Europ Assist”in order to tell about a damage of a car”. Then T3 may have a KR

*Situation (e1, telephone-call * (Agent1, certain person * (Owner, certain insur-police1))*
*(Object2, certain firm1 * (Name, “Europ Assist”)(Goal, info-transfer * (Theme1,*
*certain damage * (Object1, certain car))))* .

The considered examples show that SK-languages enable us, in particular, to describe the conceptual structure of texts with : (a) references to the meanings of phrases and larger parts of texts , (b) compound designations of sets, (c) definitions of terms , (d) complicated designations of objects, (e) generalized quantifiers ("arbitrary", "certain", etc.). Besides, SK-languages provide the possibilities to describe the semantic structure of definitions, to build formal analogues of complicated concepts, to mark by variables the designations of objects and sets of objects, to reflect thematic roles.

The creation of a semantic networking language belonging to a new generation on the basis of the definition of the class of SK-languages, in particular, will allow for: (1) constructing not only semantic representations (SRs) of separate sentences but also SRs of complicated discourses with the help of reflecting the references to the previously mentions entities and to the meanings of phrases and larger fragments of discourses; (2) forming compound designations of sets, concepts, goals of intelligent systems and destinations of things; (3) joining with the help of logical connectives “and”, “or” not only the designations of assertions (as in predicate logic) but also the designations of concepts, objects, sets of objects; (4) reflecting the semantic structure of the phrases with the words “concept”, “notion”; (5) considering non-traditional functions with arguments and/or values being sets of objects, sets of concepts, SRs of texts, sets of SRs of texts.

Thus, the theory of K-representations opens the real prospects of constructing a semantic networking language of a new generation with the expressive possibilities being much closer to the expressive possibilities of Natural Language in comparison with the language UNL described in (UNL 2005, 2006).

In (Fomichov 1996 – 2005b), the hypothesis is formulated that the theory of SK-languages provides the effective means for describing structured meanings (i.e., for representing contents) of arbitrary NL-texts in arbitrary thematic domains. That is why the following conjecture seems to be well grounded: the theory of K-representations provides a model of a Universal Semantic Networking Language.

A New Algorithm of Semantic-Syntactic Analysis of Natural Language Text

It was mentioned above that the third basic constituent of the theory of K-representations is a complicated, strongly structured algorithm carrying out semantic-syntactic analysis of texts from some practically interesting sublanguages of NL. This algorithm, called SemSyn, is 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, this algorithm is described in two final chapters of the monograph (Fomichov 2005a).

Example. Let T = “The antibiotic “Zinnat”, produced by the firm “GlaxoWelcome”, cures the maladies caused by a coccus flora”. Then the algorithm SemSyn constructs the K-representation

*(Situation(e1, producing * (Agent1, certn firm1 “ (Name1, “GlaxoWelcome”) : x1)*
(Time, current-time)(Product1, certn antibiotic “ (Name1, “Zinnat”) : x2) ∧
*Situation(e2, curing1 * (Agent1, x2)(Process1, all malady1 * (Cause,*
any flora1 “ (Kind1, coccus)))) .

An important feature of this algorithm is that it doesn’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 a complicated algorithm is completely described with the help of formal tools, that is why it is problem independent and doesn't depend on a programming system. The algorithm is implemented in the Web programming language PHP.

Conclusions

The analysis of expressive power of the language UNL provides the possibility to establish an analogy between the studies on constructing a semantic networking language (UNL being one of its versions) and the researches on the development of the advanced informational languages for forming Web-documents. The conclusion can be drawn that, similarly to the ongoing process of the transition from the language HTML to new, semantically-structured means for representing information on the Web, in the field of constructing a semantic networking language (SNL) the demands of practice must lead in the nearest years to the creation of a SNL belonging to a new generation in comparison with UNL.

The prospects of using the theory of K-representations for the elaboration of a SNL with the expressive power exceeding the expressive power of UNL are set forth. The hypothesis is put forward that the definition of the class of SK-languages can be used as a model for the development of a Universal Semantic Networking Language.

The examples considered above show that the UNL-expressions describing various entities and binary relations between these entities can be easily transformed into the K-strings (i.e. the expressions of SK-languages). That is why the linguistic processors designed in many countries under the framework of the UNL project can be easily modified in order to build K-strings. Thus, the choice of the theory of K-representations for continuing the studies in this direction is practically possible.

The principal advantages of such a choice are as follows:

- In comparison with the publications on UNL, the theory of K-representations provides a powerful and flexible framework for designing semantic-syntactic analyzers of arbitrarily complicated discourses;
- the theory of K-representations gives the formal means being convenient for building semantic representations of sentences with the descriptions of sets, compound descriptions of goals and compound destinations of things.

References

Fomichov, V. (1992); Mathematical Models of Natural-Language-Processing Systems as Cybernetic Models of a New Kind; Cybernetica (Belgium), Vol. XXXV, No. 1 (pp. 63-91)

Fomichov, V.A. (1996); A Mathematical Model for Describing Structured Items of Conceptual Level; Informatica. An International Journal of Computing and Informatics (Slovenia); Vol. 20, No. 1 (pp. 5-32)

Fomichov, V.A. (2002); 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)

Fomichov, V.A. (2004); Theory of Standard K-languages as a Model of a Universal Semantic Networking Language; Preconference Proceedings "Intelligent Software Systems for the New Infostructure" (Focus Symposium in conjunction with the 16th Intern. Conf. on Systems Research, Informatics and Cybernetics – InterSymp-2004, July 29 – Aug. 5, 2004, Germany). Focus Symposia Chair: Jens Pohl - CAD Research Center, Cal Poly, San Luis Obispo, CA, USA (pp. 51-61)

Fomichov V.A. (2005a); The Formalization of Designing Natural Language Processing Systems. Moscow, MAX Press, 368 P. (in Russian)

Fomichov V.A. (2005b); Standard K-Languages as a Powerful and Flexible Tool for Building Contracts and Representing Contents of Arbitrary E-Negotiations; K. Bauknecht, B. Proell, H. Werthner (Eds.), The 6th Intern. Conf. on Electronic Commerce and Web Technologies "EC-Web 2005", Copenhagen, Denmark, Aug. 23 - 26, 2005, Proceedings. Lecture Notes in Computer Science. Vol. 3590. Springer Verlag (pp. 138-147)

Uchida, H., Zhu, M., and T. Della Senta (1999); A Gift for a Millennium, A book published by the United Nations University; available on-line at http://www.unl.ias.unu.edu/publications/gm/index.htm

UNL (2005); Universal Networking Language (UNL) Specifications, Version 2005, www.undl.org/unlsys/unl/unl2005/

UNL (2006); Universal Networking Language (UNL) Specifications, Version 2005, Edition 2006, 30 August 2006, UNL Center of UNDL Foundation, http://www.undl.org/unlsys/unl/unl2005-e2006/

An Approach to Manage Knowledge Based on Multi-Agent System using an Ontology

Davy Monticolo[1,2], Vincent Hilaire[1], Samuel Gomes[1], Abder Koukam[1]

[1] SeT Laboratory, University of technology UTBM, 90 010 Belfort (France)
{Davy.Monticolo, Vincent.Hilaire, Abder.Koukam, Samuel Gomes}@utbm.fr
[2] Zurfluh-Feller, 25150 Roide (France)

Abstract

This paper presents a knowledge management experiment realized in an industrial company. Our research concerns the development of a knowledge engineering module integrated in a collaborative eGroupware system. This platform is used by engineers to realise their projects in a collaborative way and in following a defined professional process. The first step of our approach is based on the modelling of the professional process used by professional actor. We have developed a formalism called RIOCK (Role Interaction Organisation Competence and Knowledge) to identify the emanating Knowledge resulting from the interaction between the roles played by professional actors. According to the obtained cartography of Knowledge, we have defined a typology of Knowledge and built an ontology to create a representation language in order to share and broadcast Knowledge. In other hand, the RIOCK models allow us to design a knowledge engineering module based on a multi-agent system. This system monitors the action of the professional actors inside the eGroupware and capitalizes, annotates, and broadcasts Knowledge in using the semantic web technologies and the ontology.

Keywords

knowledge engineering, multi-agent system, ontology.

1 Introduction

In recent years, manufacturing companies have to innovate in order to improve competitiveness and business performance. They must bring innovative products to market more effectively and more quickly to maximize customer interest and sales. Thus project teams of engineering departments have to leverage and reuse the product-related intellectual capital. The design process has to be rationalized in managing knowledge, know-how and technological patrimony. Moreover the information technology explosion of the last decade, allows to companies to manage efficiently masses of information, with powerful search capabilities i.e. catalogues and on-line information systems. However engineers have at their disposal too much information and prefer to use their experience rather than these information systems.

This article describes works carried out in collaboration with project teams of the Zurfluh-Feller company specialized in the mechanisms of rolling shutters. Our investigation is about using several emerging technologies to support a Knowledge Engineering approach. This one produces methods, techniques and platforms to collect, analyze, structure and represent

Knowledge. This Knowledge arises from collaborative professional activities testifies to a relative truth (Matta et al. 2000). This knowledge is based on experiences of human resources, and project experiences in terms of project management issues, design technical issues and lessons learned. The coherent integration of this dispersed know-how in a corporation, aimed at enhancing its access and reuse, is called "corporate memory" or "organizational memory". It is regarded as the central prerequisite for IT support of Knowledge Management and is the means for knowledge conservation, distribution, and reuse. We work on the build of a project memory model we have called 'MemoDesign'. A project memory is simply organizational memory for a project team; it has a more limited scope than organizational memory since it is composed by the knowledge emanating only during engineering projects.

Our approach aims to analyze and model the professional process used by project team in order to identify emanating Knowledge. In collaboration with project teams, we have defined the types of knowledge to capitalize and which represent the structure of our project memory. Then we have explained and described each concept and relation of the project memory to build the associated ontology (Called OntoDesign).

To manage this project memory, we have developed agents to constitute a multi-agent system (MAS) i.e. a loosely coupled network of agents that work together as a society. A MAS is heterogeneous when it includes agents of at least two types. We have defined several types of agent to support each part of the knowledge management process in order to assist engineers to exploit Knowledge all along projects.

The first part of this paper presents the related works in agents systems in knowledge engineering. The second part presents our building process of a project memory with its associated ontology. The last section describes the multi-agents system dedicated to the management of project memories.

2 Related Work

Knowledge Engineering aims to collect, analyze, structure and represent Knowledge. Knowledge environments can be seen as distributed systems where actors with different specialties share their know-how in order to achieve their professional activities. In such environments the choice of multi agents system is motivated by the following functionalities:

- To manage the heterogeneous and distributed Information,

- To solve complex problems in split them,

- To provide a assistant to professional actors in reusing Knowledge,

Agents support and extend human interactions and capabilities, organize knowledge by facilitating information and documents retrieving and cataloguing. A new and very interesting research field recently growing is Agent-Mediated Knowledge Management (AMKM) whose intent is to link the Knowledge Engineering theories with the Agent-based models. Dignum (Dignum et al.2005),Van Elst and Abecker (Abecker et al. 2003) argued that "the basic features of agents (social ability, autonomy and proactiveness) can alleviate several of the drawbacks of the centralized technological approaches for KM". They proposed three dimensions to describe agent KM systems: the system development layer (from analysis to implementation of the system), the macro-level structure of the system (single agent and

Multi-agents models) and the KM applications area (share, learn, use, distribution and so on). Taking into account the second dimension, they proposed a classification of software or experimental models of agent systems that could support Knowledge Engineering. For example, agents whose task is to support knowledge workers in their daily work in a way to become "a personal assistant who is collaborating with the user in the same work environment" (Maes 1994). Many different examples fall into this category, like Lieberman's Letizia (Lieberman 1995), the OntoBroker developed by Staab and Schnurr (Staab and Schnurr 1995) and the AACC project in (Enembreck and Barthès 2002).

Agent-based systems have also been developed to support the creation of Organizational Memories, "meta-information systems with tight integration into enterprise business processes, which relies on appropriate formal models and ontologies as a basis for common understanding and automatic processing capabilities" (Abecker et al. 2003). An example can be found in the FRODO project, a Distributed Organizational Memory System in which agents communicate with the FIPA ACL language. In this work, agents are not only described by their knowledge, goals and competencies, but also by their rights and obligations. In the same aim, Gandon (Gandon et al. 2002) develops a MAS using semantic web technologies to manage organizational memories.

3 Process building of a project memory and its ontology

The cartography of knowledge makes it possible to detect knowledge to capitalized in a project. We refer to the definition given by Speel (Speel et al. 1999) explaining that the cartography of knowledge is a whole of processes, methods and tools to analyze knowledge in order to discover their significances and to visualize them in a comprehensible form. In (Ermine et al. 2005), the purpose of a cartography of knowledge is to provide a structuring of the cognitive resources of the organization. They define three approaches to organize these resources: procedural classification (according to the processes of the company), functional classification (usually with a flow chart), and conceptual classification or by fields (information organized in subjects, objects and finalities). Thus the cartography of knowledge requires a precise analysis of the processes used in the company in order to determine knowledge to preserve, develop, or give up. The cartography becomes a decision making tool

Our cartography of knowledge is inspired by the step presented by (Suyeon and Hyunseok 2003) with four steps; to identify knowledge inside professional processes, to define knowledge necessary to capitalize, to describe the typology and taxonomy of knowledge and to specify the attributes and relations of the knowledge concepts.

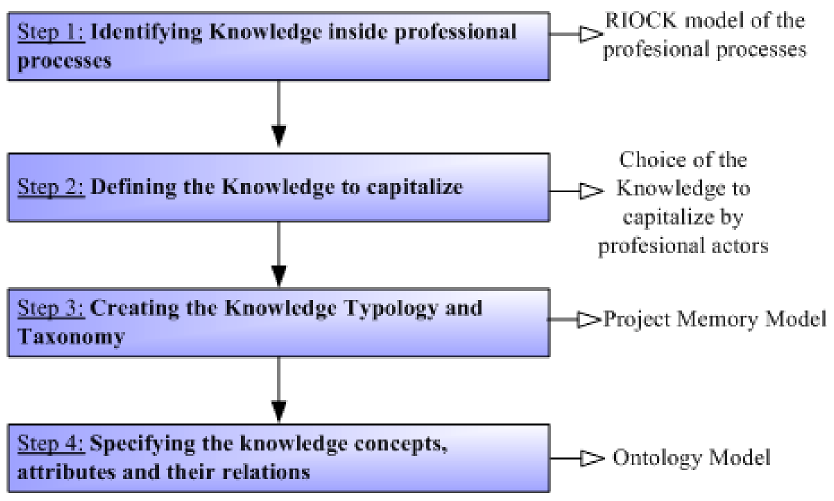

Step 1: Identifying Knowledge inside professional processes
RIOCK model of the profesional processes
Step 2: Defining the Knowledge to capitalize
Choice of the Knowledge to capitalize by profesional actors
Step 3: Creating the Knowledge Typology and Taxonomy
Project Memory Model
Step 4: Specifying the knowledge concepts, attributes and their relations
Ontology Model

Fig.1: Process to realize the Knowledge Cartography

3.1 Step 1: Identifying Knowledge with the formalism RIOCK

Our cartography of knowledge is based on its identification from an organisational approach to model the professional processes implemented in projects. Our modelling is built with the concepts of Roles, Interactions, Organization Competence and Knowledge (Hilaire and al. 2000), (Monticolo et al. 2007). An organization models the professional process containing several under organizations modelling themselves the phases and activities of the process. Inside, roles are generic behaviours. These behaviours can interact mutually according to interaction pattern. Such a pattern which groups generic behaviours and theirs interactions constitutes an organization. Indeed professional actors instantiate an organization (roles and interactions) when they exhibit behaviours defined by the organization's roles and when they interact following the organization interactions (Castelfranchi 2004).

Inside the professional process, the actors use and share their knowledge to carry out in a collaborative way the activities of engineering and thus develop their learning resulting from the knowledge capitalization process. From the experiments and observations carried out in the company, we defined, for each organization (activity of the professional process) several roles interpreted by professional actors. We allocate to these roles the competences which they use to achieve the activities. The competence is the capacity for an individual to implement its knowledge and to develop its know-how. Competences are also developed during the achievement of professional activities, in which the share of knowledge takes place. Each competence is aggregated with a set of knowledge.

In the activity (i.e. organization) 'to write the schedule of conditions' we observe three roles (Figure 2). The role 'Technical commercial assistant' uses one of its competences, we read it like *the capability to* 'To formalize the requirement of the customer'. This competence requires three elements of Knowledge which are used to satisfy the organization. In the RIOCK diagram the type of knowledge are read like *Knowledge on*, for example the role project leader possesses the Knowledge on 'means of industrialization of the company'. In addition RIOCK presents the result of the collaboration among these three roles; here this is the schedule of condition.

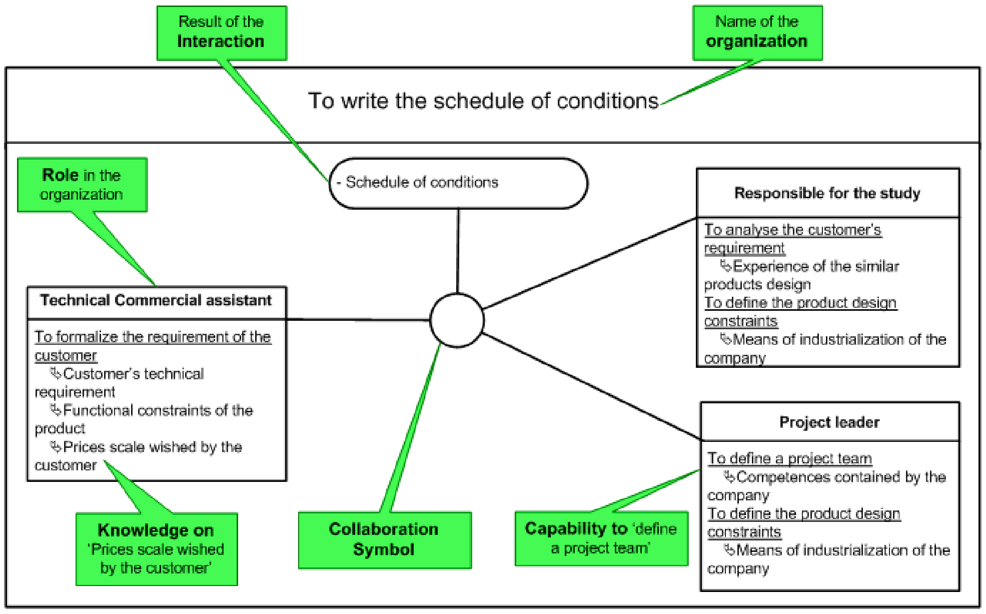

Result of the Interaction
Name of the organization
To write the schedule of conditions
Role in the organization
- Schedule of conditions
Responsible for the study
To analyse the customer's requirement
Experience of the similar products design
To define the product design constraints
Means of industrialization of the company
Technical Commercial assistant
To formalize the requirement of the customer
Customer's technical requirement
Functional constraints of the product
Prices scale wished by the customer
Project leader
To define a project team
Competences contained by the company
To define the product design constraints
Means of industrialization of the company
Knowledge on 'Prices scale wished by the customer'
Collaboration Symbol
Capability to 'define a project team'

Fig.2: RIOCK model for the stage 'To write the schedule of conditions'

3.2 Step 2: Defining knowledge to capitalize by the professional actors

The modelling of the professional process with RIOCK allows us to obtain a precise identification of knowledge used, shared and created. With this modelling we have wrote a series of knowledge which we have submitted to the professional actors. Afterwards they have validated knowledge to capitalize in the project memory. We can draw up a list of knowledge identified like necessary to capitalize.

3.3 Step 3: Creating the Knowledge Typology and Taxonomy

In order to present a Knowledge classification we have realised a regrouping of Knowledge. On the whole Knowledge identified like necessary to capitalize, we have six groups of Knowledge (table 1). Each group represents a type of knowledge.

Name of the knowledge type	*Knowledge*
Context of the projet (ProjectContext)	- Knowledge presenting the origin of the project - Knowledge describing the organization of the project
Evolution of the project (ProjectEvolution)	- Knowledge related to the history of the evolution of the project
Professional processes set up in the project (ProjectProcess)	- Knowledge presenting the activities carried out, the interventions of the professional actors and the information handled for each activity
Glossary of the projet (ProjectGlossary)	- Knowledge defining the vocabulary used during the project
Expertise in the project (ProjectExpertise)	- Knowledge related to the professional rules used to develop the product
Exprrience developed in project (ProjectExperience)	- Knowledge describing the errors, failures and difficulties in the project

Table 1 : Knowledge regrouping

In order to structure the Knowledge, we have created a taxonomy. This is a classification of information entities in the form of a hierarchy, according to the presumed relationships of the real-world entities that they represent. Furthermore the classification is based on the similarities of the information entities called concepts. This taxonomy represents the structure of the project memory MemoDesign.

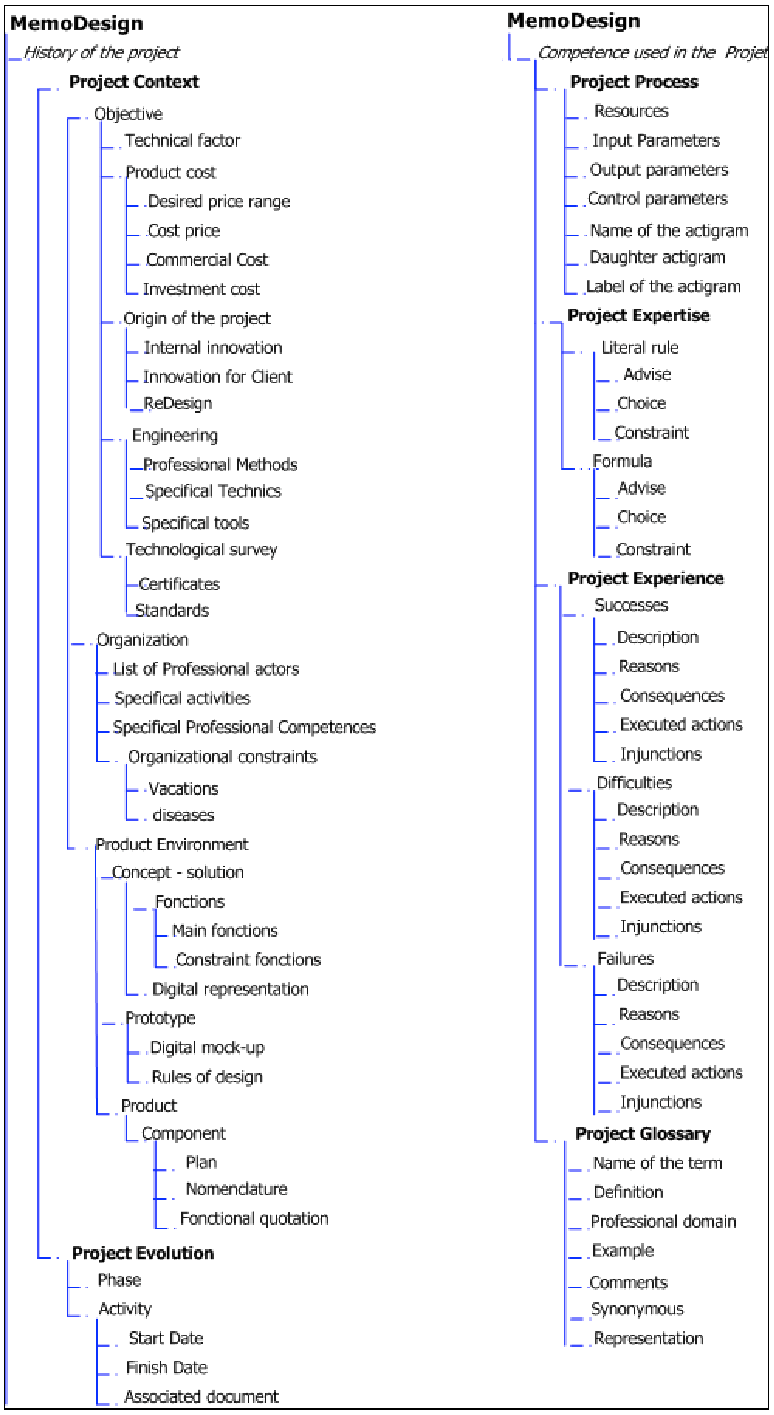


Fig.5: Knowledge Taxonomy of MemoDesign

3.4 Step 4: Specifying the Knowledge concepts, attributes and their relations

The first part of this step is to define the relations between the concepts of the taxonomy. We give a unique name of potential relations the concepts they link in specifying domain and range. The second part of this step is to associate a set of attributes for each concept. The figure 6 presents a graphical view of the ontology OntoDesign with the relations, attributes and concepts associated of the project evolution and the project process.

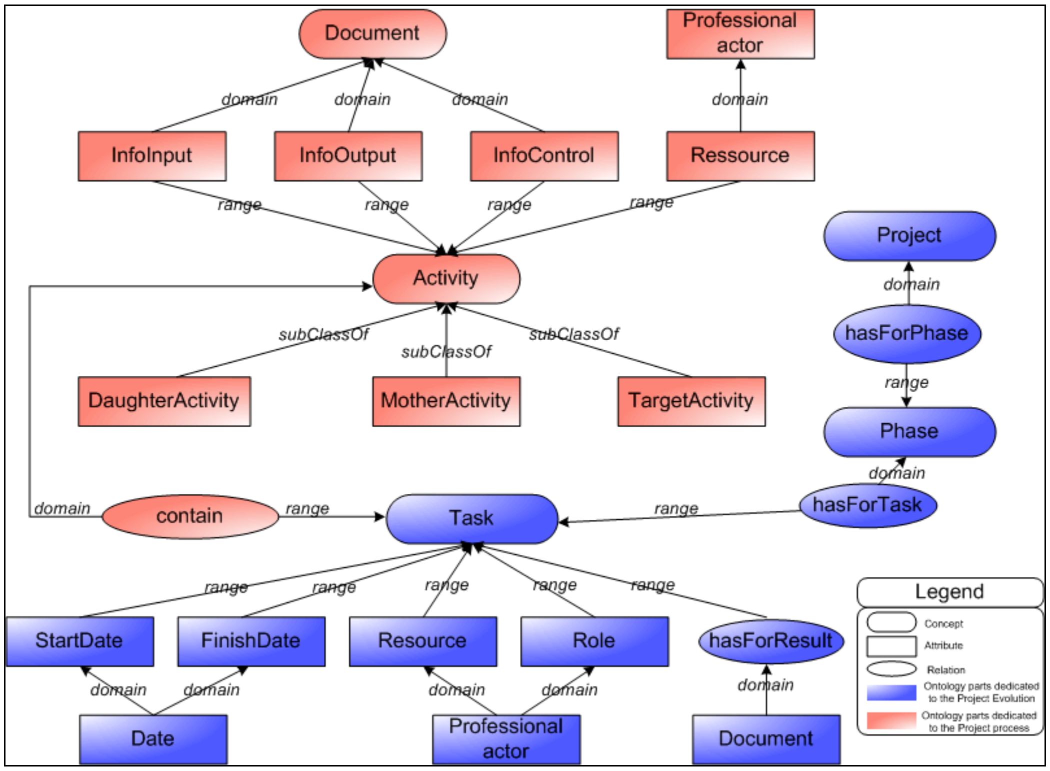
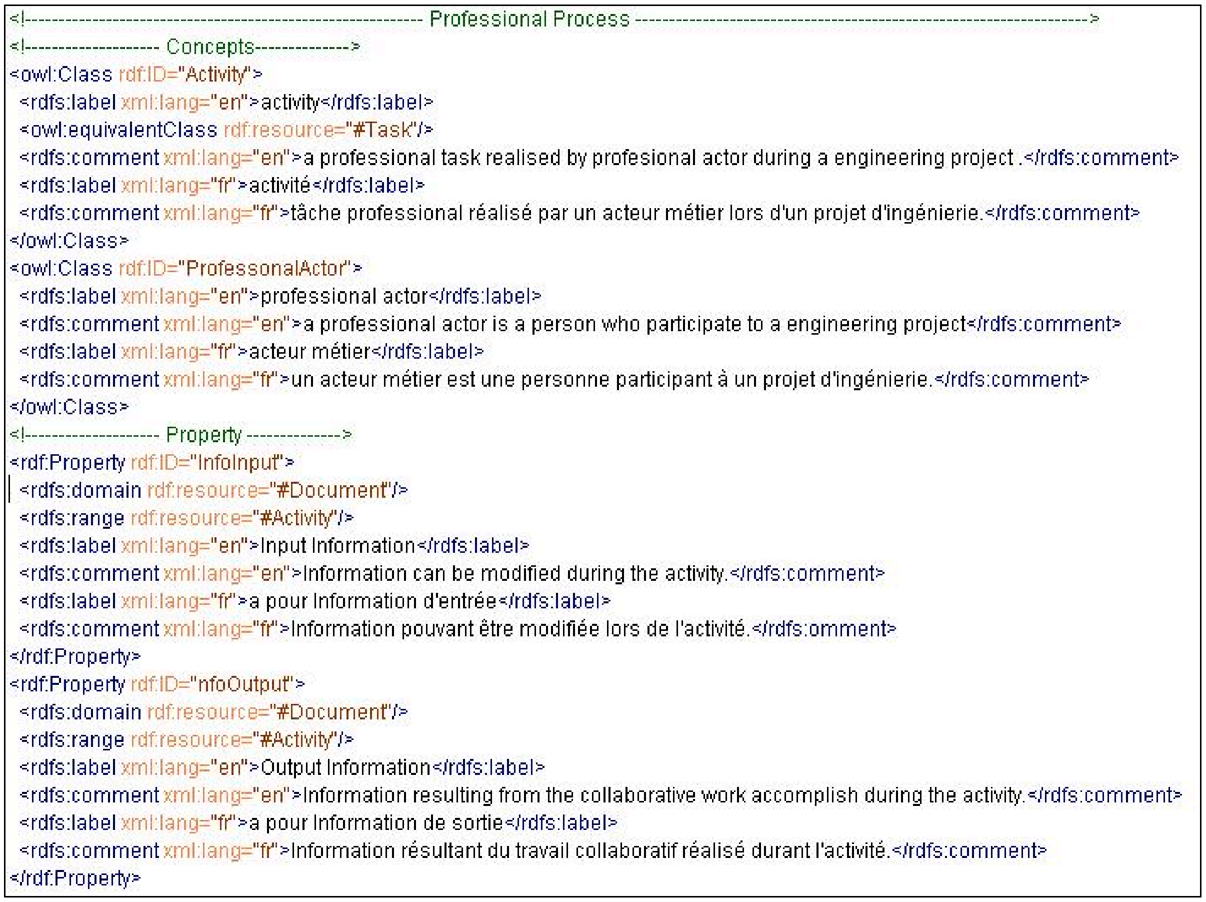


Fig.6: An extract of the ontology OntoDesign

3.5 Using the semantic Web language to develop the ontology OntoDesign

We specify the project memory concepts and their relationships in the ontology OntoDesign and formalize this ontology in OWL using the W3C recommendations. OntoDesign provides an integrated conceptual model for sharing information related to a mechanical design project.

An OWL property (figure 7) is a binary relation to relate an OWL Class (Concept in OntoDesign) to another one, or to RDF literals and XML Schema datatypes. For example, the “infoInput” property relates the *Document* class to the *Activity* class. Described by these formal, explicit and rich semantics, the domain concept of *Activity*, its properties and relationships with other concepts can be queried, reasoned or mapped to support the Knowledge sharing across the mechanical design projects.



```
<!------------------------------------------------------------ Professional Process ----------------------------------------------------------------->
<!------------------- Concepts------------->
<owl:Class rdf:ID="Activity">
  <rdfs:label xml:lang="en">activity</rdfs:label>
  <owl:equivalentClass rdf:resource="#Task"/>
  <rdfs:comment xml:lang="en">a professional task realised by profesional actor during a engineering project .</rdfs:comment>
  <rdfs:label xml:lang="fr">activité</rdfs:label>
  <rdfs:comment xml:lang="fr">tâche professional réalisé par un acteur métier lors d'un projet d'ingénierie.</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="ProfessonalActor">
  <rdfs:label xml:lang="en">professional actor</rdfs:label>
  <rdfs:comment xml:lang="en">a professional actor is a person who participate to a engineering project</rdfs:comment>
  <rdfs:label xml:lang="fr">acteur métier</rdfs:label>
  <rdfs:comment xml:lang="fr">un acteur métier est une personne participant à un projet d'ingénierie.</rdfs:comment>
</owl:Class>
<!------------------- Property -------------->
<rdf:Property rdf:ID="InfoInput">
| <rdfs:domain rdf:resource="#Document"/>
  <rdfs:range rdf:resource="#Activity"/>
  <rdfs:label xml:lang="en">Input Information</rdfs:label>
  <rdfs:comment xml:lang="en">Information can be modified during the activity.</rdfs:comment>
  <rdfs:label xml:lang="fr">a pour Information d'entrée</rdfs:label>
  <rdfs:comment xml:lang="fr">Information pouvant être modifiée lors de l'activité.</rdfs:omment>
</rdf:Property>
<rdf:Property rdf:ID="nfoOutput">
  <rdfs:domain rdf:resource="#Document"/>
  <rdfs:range rdf:resource="#Activity"/>
  <rdfs:label xml:lang="en">Output Information</rdfs:label>
  <rdfs:comment xml:lang="en">Information resulting from the collaborative work accomplish during the activity.</rdfs:comment>
  <rdfs:label xml:lang="fr">a pour Information de sortie</rdfs:label>
  <rdfs:comment xml:lang="fr">Information résultant du travail collaboratif réalisé durant l'activité.</rdfs:comment>
</rdf:Property>
```

Fig.7: Segment of the OWL code for specify the Project Process

4 KATRAS, a Multi Agents System dedicated to the management of project memories

We propose to introduce a multi agents based model to provide the cognitive and social approach in modelling the intelligent collective and individual behaviours which composed the mechanical design process. With the RIOCK formalism (section 3) we have observed that professional actors play different roles in different organizations (activities) and for each of them they develop competences in using knowledge.

With regard to these concepts, we propose a Multi Agents model called KATRAS (Knowledge Acquisition Traceability Re-used Agents System) which the aim is to capitalize from the roles of the professional actors all along projects.

This MAS architecture is based on three levels:

- The first level ensures the traceability of users activities inside an e-groupware plate-form. In this level we find the type of agents called `Professional Agents'. These agents exist for one project. They monitor roles of the professional actors through-out projects in building their own RIOCK organization to identify emanating Knowledge. Their objective is to ensure a traceability of the collaborative actions carried out by professional actors in order to capture and annotate knowledge. They consult the ontology OntoDesign to use a common vocabulary in their annotations.

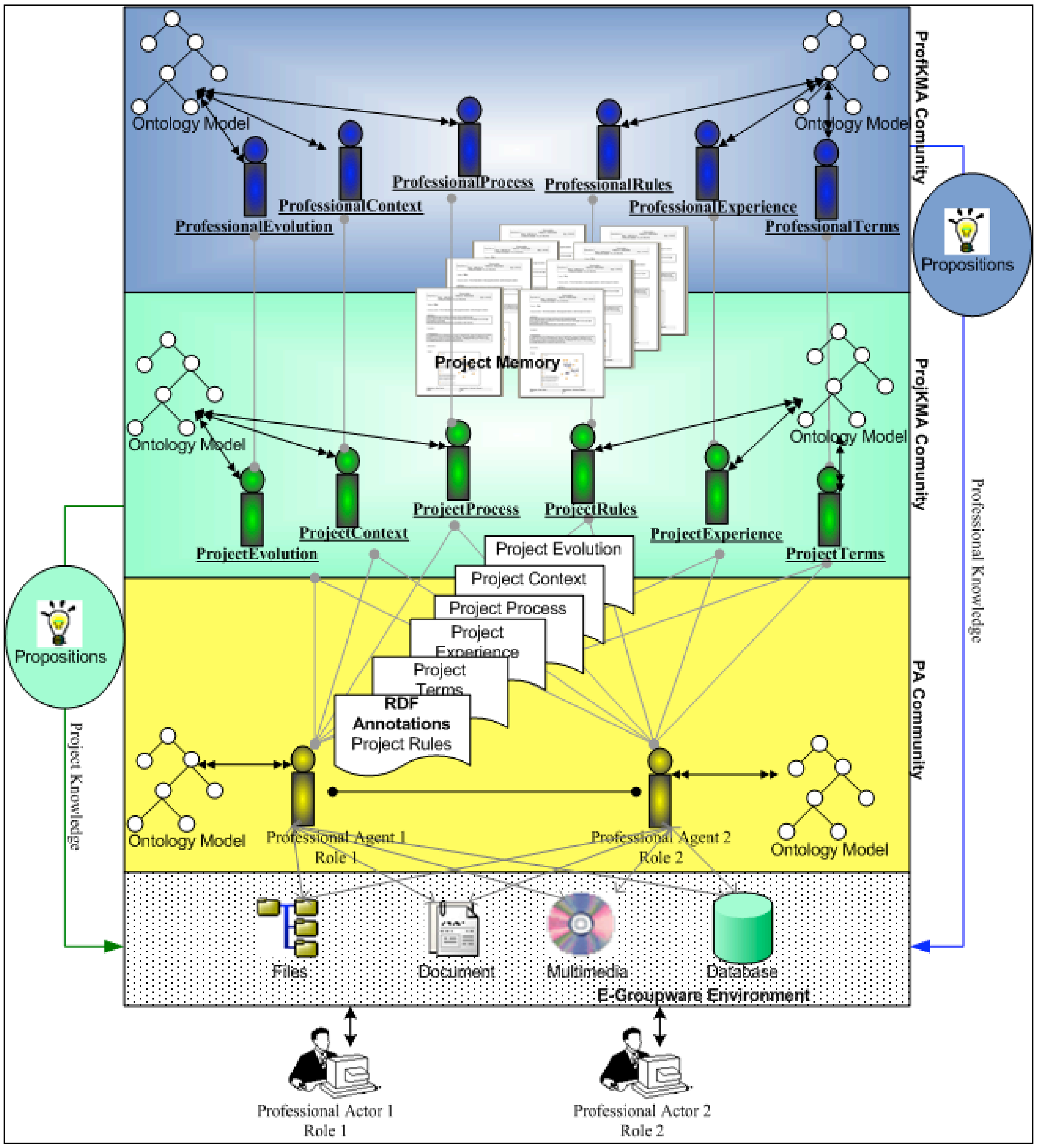


Fig.8: Segment of the OWL code for specify the Project Process

- The second level gathers mechanisms of Knowledge capitalization. In this level we find the type of agents call `Project Knowledge Managers Agents' (i.e. ProjKMA). The aim of ProjKMA is to capitalize from knowledge annotations of engineering activities communicated by the PA agents. The organization of this Knowledge is done according to the ontology model OntoDesign presented in section. Communities of ProjKMA exist for each project. Therefore the ProjKMA build the project memory of the project in which they are created. They also propose solution to professional actors all along the current project.

- The third level contains the agents type ‘Professional Knowledge Managers Agents’ (ProfKMA). These agents exist for every projects, their aim is to synthesize the Knowledge structured according to project memories for the whole of projects. The Knowledge capitalized and reused during one project is a Project Knowledge, and the Knowledge capitalized during the whole of the projects and reused in a new project is a Professional Knowledge. Therefore ProfKMA are able to propose solution from professional Knowledge all along a new project.

Conclusion

In this article, we present our approach to cartography Knowledge and to build a project memory and its associated ontology. The first step of this methodology is based on the use of the RIOCK formalism to identify Knowledge. This Knowledge has to be validated by professional actors. Then with the list of validated knowledge we can create a typology and a taxonomy. This work allows building the structure of a project memory called MemoDesign. The last step of this methodology is about the specification of concepts, attributes and their relation in order to build the ontology called OntoDesign. The ontology provides an integrated conceptual model for sharing information. We showed that the ontology is a keystone of a multi-agent project memory system. This Agent model called KATRAS is based on an organizational model to support the Knowledge Management Process. Agents monitor the role of professional actors for each professional activity. They are able to research, identify and annotate the Knowledge resulting from activities. Afterwards they capitalize them in project memories. Thus project memories become a Knowledge repository used to handle Knowledge and to propose solutions to the professional actors when they carry out new activities. Nowadays this agent model is under development in the company. Agents are implemented in the Madkit Platform (Gutknecht and Ferber 200) and used the java jena API to perceive the OWL ontology. The first results propose a assistance to professional actors in the following domains:

- Project Management with the history (Lateness reasons and effects) of the professional activities and the proposition of a project model.
- Product management with the contribution of professional rules used during the development of similar product.
- Process management where agents propose optimized professional process from the sequences of encountered activities.

As these application deployments progress, we believe we can reduce the design time of routine products. In other hand, the contribution of our agents to develop novelties product intervenes at the time of the solutions research and avoid redeveloping existent products.

References

Abecker A., Bernardi A., and Van Elst L. (2003); Agent technology for distributed organizational memories. In Proceedings of the 5th International Conference On Enterprise Information Systems, Vol. 2, pages 3–10.

Castelfranchi C. (2004); Engineering Social Order, Fifth International Workshop Engineering Societies in the Agents World, Toulouse, France

Dignum V., Dignum F., Meyer J.J. (1005): An Agent-Mediated Approach to the Support of Knowledge Sharing in Organizations. Knowledge Engineering Review, Cambridge university Press, 19(2), pp. 147-174

Enembreck F., Barthès J.P. (2002); Personal assistant to improve CSCW. In Proceeding of CSCWD, Rio de Janeiro, 2002.

Ermine J-L., Boughzala I., Tounkara T. (2005); Using Cartography to sustain Inter-Generation Knowledge Transfer : The M3C Methodology, ICICKM 2005, DUBAI,

Gandon F., Poggi A., Rimassa G., Turci P. (2002); Multi-Agent Corporate Memory Management System, In Engineering Agent Systems: Best of "From Agent Theory to Agent Implementation (AT2AI)-3", Journal of Applied Artificial Intelligence, Volume 16, Number 9-10/October - December 2002, Taylor & Francis, p699 – 720

Gutknecht O., Ferber J. (2000); Madkit : A generic multi-agent platform, AGENTS'00 : 4th International Conference on Autonomous Agents, Barcelona, Spain

Hilaire V., Koukam A., Gruer P. & Müller J-P (2000); Formal Specifi-cation and Prototyping of Multi-Agent Systems. Engineering Socie-ties in the Agents' World, in Lecture Notes in Artificial Intelligence. n°1972, Springer Verlag.

Lieberman H. (1995);. Letizia: An agent that assists web browsing. In Chris S. Mellish, editor, Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, Quebec, Canada, August 1995.

Maes P. (1994); Agents that Reduce Work and Information Overload. In Communications of the ACM, Vol. 37, No.7

Matta N., Ribiere M., Corby O., Lewkowicz M., Zaclad M. (2000); Project Memory in Design, Industrial Knowledge Management - A Micro Level Approach, Rajkumar Roy (Eds), Springer-Verlag

Monticolo D., Gomes S., Hilaire V., Serrafero P. (2007); 'Knowledge capitalization process linked to the design process', International Join Conference on Artificial Intelligence (IJCAI). Workshop on Knowledge Management and Organisational Memories, Hyderabad-India, p13

Speel, P., Shadbolt, N., De Vries, W., Van Dam, P. (1999); “Knowledge map making for industrial purpose”, paper presented at Conférence KAW99, Banff

Staab S. and Schnurr H.-P. (2000); Smart task support through proactive access to organizational memory. Knowledge–based Systems, 13(5):251–260.

Suyeon K. Euiho S. and Hyunseok H. (2003); “Building the knowledge map; an industrial case study”, Journal of Knowledge Management, vol7 n°2, pp 34-45

Agent Facilitated System to Support Conceptual Architectural Design as a Reflective Conversation

Geeta Arjun and Jim Plume

Faculty of the Built Environment
University of New South Wales, Australia

Abstract

This paper describes the research undertaken to investigate and develop an agent facilitated design conversation system to aid creative thinking in the early stages of architectural design. Architectural design is comprised of different artistic, functional, material, ecological etc, demands, which may be inconsistent, combined through architectural design in a novel way (Haapasalo, 2000). As such, architects need to think from different perspectives for any current design situation. Cognitive studies in the design processes have gained considerable significance in the past decade with a range of methods employed to study the designer's mind. Building on these studies, a prototype agent-based system has been designed and implemented to assist the designer in the conceptual stages of design. The prototype is aimed at triggering ideas through stimulating the designer's experiential memory. This paper describes the framework and implementation of the prototype, which is currently in progress.

Keywords

conceptual design; design cognition; design conversation; agent systems

Introduction

Increasing research in design cognition has led to the development of computational models in artificial intelligence that are founded upon cognitive processes. Over the years, we have seen the computer as a draftsman, modeler and evaluator. The applications suggest significant potential in the use of computers in architectural design. However, the process of trying to develop useful roles for the computer in the design process is seen as a yet unfinished journey (Lawson, 1997). The aim of this research is to impact the stage of idea development in the conceptual stages of design, wherein the prototype acts as a catalyst for innovative idea generation. Based on Valkenburg and Dorst's (1998) paradigm on team-designing, a computational framework is proposed and implemented to assist the designer in the early stages of design. The designer converses with a team of software agents to recognize and analyse the possibilities of concepts for specific design situations.

Background

The early stages of architectural design

In the early stages of the design process, architects are constantly producing drawings meant for their own understanding. This process has been beautifully described by Schön (1995) as the 'designer having a reflective conversation'. A concept forms the foundation for ideas in the early stages of the architectural design process. A building is appreciated because of its concept, its meaning, its underlying and integrating idea, which gives it an added value with regard to the commonplace (Heylighen et al, 1999). The design conversation during concept formation continues until the designer is satisfied with the result produced.

Design cognition and creativity

Studies that compare the relationship between cognition and design processes have gained considerable significance over the past decade. A driving force behind this work has been its potential to identify the abilities of reasoning and creativity (Oxman, 1996).

Associative thinking in design

It is commonly accepted that one concept leads to another through a process of triggering groups of associated information. Gruber (1980, cited in Goldschmidt, 2005) states that "Interesting creative processes almost never result from single steps, but rather from concatenations and articulation of a complex set of interrelated moves." Studies have shown that good ideas are those that spin the largest number of links among themselves and other ideas. The interconnectivity of design domains in the conceptual stages of architectural design has been considered as an aspect of Associational Thinking. As derived from Schön's (1995) example of moves, a designer works in one domain. It is, however in the other domains that the designer discovers the unintended consequences and qualities of the design move. At the point of conceiving a design, a designer is not aware of all the domains that would be affected. Because of our limited information processing capacity, we cannot, in advance of making a particular move, consider all the consequences and qualities we may eventually consider relevant to its evaluation (Schön 1992). The ability, while designing, to simultaneously consider different domains increases with experience. Particular views of the connectivity of design domains are inherent in design communities, and from their perspective, help to distinguish experts from novices (Schön, 1992). The relationship or connectivity between different domains gets restructured as individual designers gather more experience. This restructuring of relationships increases the scope of complexity in the design process itself, thereby opening up new avenues of thinking in the design process.

Experiential memory

There is no simple explanation for the meaning of the term 'experience'. It generally refers to accumulated knowledge over a person's lifetime. Experience can be seen as a basis for understanding the knowledge which is present, independent of the designer's beliefs. Studies in artificial intelligence and the cognitive sciences cover the aspect of learning processes in the human mind (Suwa et al, 2000). One of the principal research issues in design cognition has been memory and the human thought process. Experiential memory is able to store events easily, but they are often more difficult to recall. It requires a trigger that can stimulate the memory to recall the experience. Once recalled, however, the person is able to remember events in some detail. Lawson (2001) describes the significance of experiential knowledge to design, especially in the relationship between design problems and solutions.

Words in the design process

Studies have shown that verbalization holds a fundamental strength in conceptual designing, not only in human communication but also in the process of thought (Jonson, 2005; Lawson and Loke, 1997). Medway (2003) describes an architectural discourse: 'Architectural discourse is pervaded by metaphor and a lot of metaphors come from language, so we talk about the 'vernacular of the building', the 'vocabulary of the building', 'buildings making statements', and 'reading buildings'. Very often designers communicate ideas over drawings by gestures or by just moving the pencil over the drawing paper. These instances show us that words form an integral part of communication and development of ideas. Architects work with words or texts that are products of linguistic choices and construct reality in particular ways (Markus et al, 2002). Language provides new perspectives on a design situation. The first body of text that the architect uses in a design project is the 'design brief', wherein the client communicates his or her ideas to the architect. This marks the beginning of the design conversation, with written words communicating assumptions or possibilities for the design outcome, as envisioned by the client. Words evoke subtle meanings and interpretations when used in the early stage of the design process (Segers, 2005; Lawson and Loke, 1997; Suwa et al, 2000). They trigger the memory and provide clues and associations for the current design situation. Menezes and Lawson (2006) study the cognitive action of 'perception' in sketches with novice and advanced designers and find that the designers attach more relevance to the 'interaction' that happens with the medium in the conceptual design stages rather than the medium itself. New thoughts emerge in this dialogue.

Approach – The Proposed Prototype

Taking insights from Lawson's design conversation system (Lawson and Loke, 1997), a proposal has been made for the application based on agent facilitated blackboard architecture to support the discursive process for triggering a dynamic association of ideas. The prototype is proposed as a dialogue between the user and domain-specific computational agents. The dialogue is aimed at triggering the experiential memory of the user and associating significant experiences from different domains of the design problem to stimulate creative thinking. Going by its objective, the prototype is named 'Design Thinker'. The mechanism for the design conversation has been adapted from Valkenburg and Dorst's (1998) model on team designing. This model represents 4 different design activities- *naming*, *framing*, *moving* and *reflecting*.

This style of conversation begins by *naming* an important aspect of the design brief that then becomes the focus or theme of the conversation. This is followed by *framing* one or more design issues that are relevant to the chosen design aspect. Once the design issues are framed and a suitable design issue is chosen, it is the beginning of a conversation between the designer and domain agents. This forms the *moving and reflecting* phase of the conversation that continues till the designer is satisfied. This phase can be compared to a brainstorming session between designers wherein different perspectives are provided for a design situation. In creating the prototype for a collaborative design conversation, a conceptual framework, as in Figure 2, has been laid out to define the important components of the system. The system is based on a *Blackboard Architecture* in which a blackboard forms a temporary database of user-selected focus terms. The blackboard is an internal system mechanism and is not viewed directly by the user. At the front end of the system is an explorer type of interface through which the user interacts with the system.

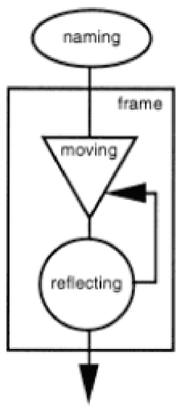


Figure 1: The mechanism of reflective practice: the four design activities and their interplay (Valkenburg and Dorst, 1998)

The knowledgebase

At the core of the agent system is the knowledgebase or ontology that forms the basis for agent reasoning. The ontology is adapted from the book, 'The Metapolis Dictionary of Advanced Architecture.' This dictionary defines the practice of architecture in a contemporary perspective by giving definitions and meanings to terms with what has come to be called *Advanced Architecture*. The dictionary defines terms which we call Knowledgebase Terms (KB Terms) with zero or more definitions provided by the different architectural authors. Therefore, each definition can be understood to provide a different architectural perspective for each KB Term. Apart from definitions, each KB Term has 3 sets of associations – Ideological, Semantic and Related associations. An ideological association links a key architectural term to zero or more groups of analogical associations, each consisting of KB Terms. A semantic association is used when a KB term is close enough to be explained through the definition of another term. A related association provides a list of terms that are related to the KB Term. In this manner, the dictionary provides the ontology with word associations that are integral to the aims of this research.

Design brief and design consideration

The prototype begins with a new project. The user enters a Project Name and specifies a Design Brief as a textual piece, written by the user, pertaining to the current project. The text for the Design Brief can be a set of requirements or a short story/paragraph describing the project. The Design Consideration is an aspect of the design brief that forms the theme or focus of the conversation. The designer can choose to change the Design Consideration at any further point during the course of the conversation.

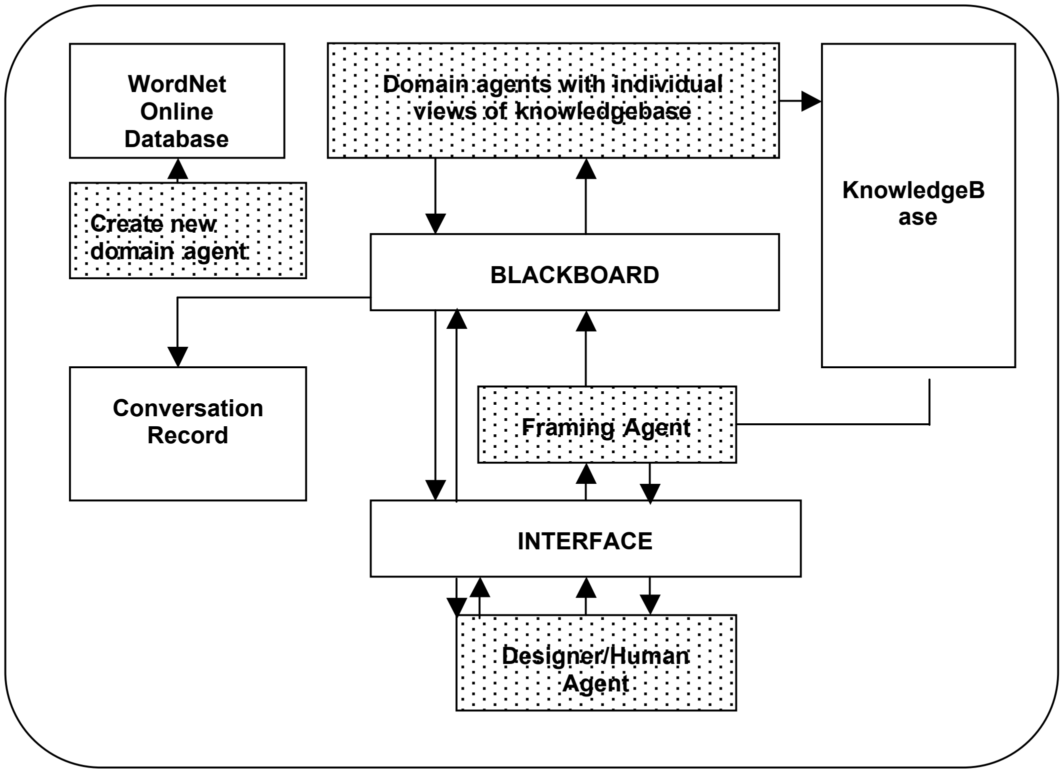


Figure 2: System architecture of the proposed agent collaboration

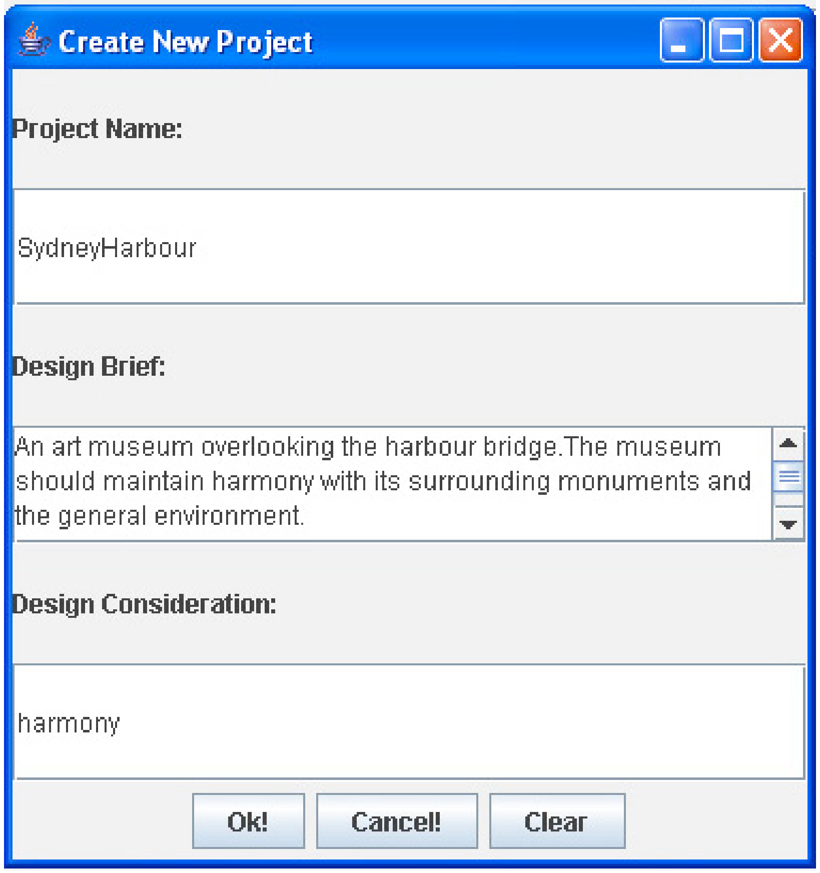
Agent types

Following the paradigm for the design conversation, the prototype is composed of 3 kinds of agents – Framing Agent, Domain Agents and User Agents.

- Framing Agent: The task of the framing agent is to identify Candidate Ideas from the knowledgebase that are significant to the Design Consideration.

- Domain Agents: Domain agents belong to a particular domain and respond to the Focus Term on the blackboard based on their View of the knowledgebase. A View is a subset of the entire structure of the main knowledgebase and forms the 'nature' of the agent. For the purpose of the prototype, we define domain agents from three domains of architectural design – Form, Space and Style.

- User Agents: Users of the prototype can create their own domain agents by providing a 'name' for the agent and a set of characteristics that are unlimited. The greater the number of characteristics, the more extensive is the agent's view of the knowledgebase.



Project Name:

SydneyHarbour

Design Brief:

An art museum overlooking the harbour bridge.The museum should maintain harmony with its surrounding monuments and the general environment.

Design Consideration:

harmony

Figure 3: The 'Create New Project' window

The Design Thinker – implementation and working

The Design Thinker interface for the conversation record is designed on the lines of an 'explorer' interface.

The left side of the interface represents the hierarchy of the program structure beginning with the meta-level, 'MyDesignWorld' containing folders for System Agents, User Agents and Projects. The 'Projects' folder lists out the projects that have been created in the system. Each project consists of a text file for the Design Brief and the Design Consideration/s that the user has defined for the project.

Phase 1:

The first phase of the conversation involves creating a project folder which consists of the project brief and a design consideration. On selection of the brief through a double click, the right pane of the interface displays the text of the brief. On selection of the *Design Consideration*, the framing agent is activated and generates a set of *Ideas* from the main knowledgebase in batches of three. The user can probe for more *Ideas* by clicking on the 'Next Idea' button on the menu bar. The user chooses one *Idea* by a double-click and it gets placed on the blackboard. This moves us to the next phase of the conversation record which is the beginning of the design conversation. The domain agents, each with their *View* of the knowledgebase, get activated in this phase.

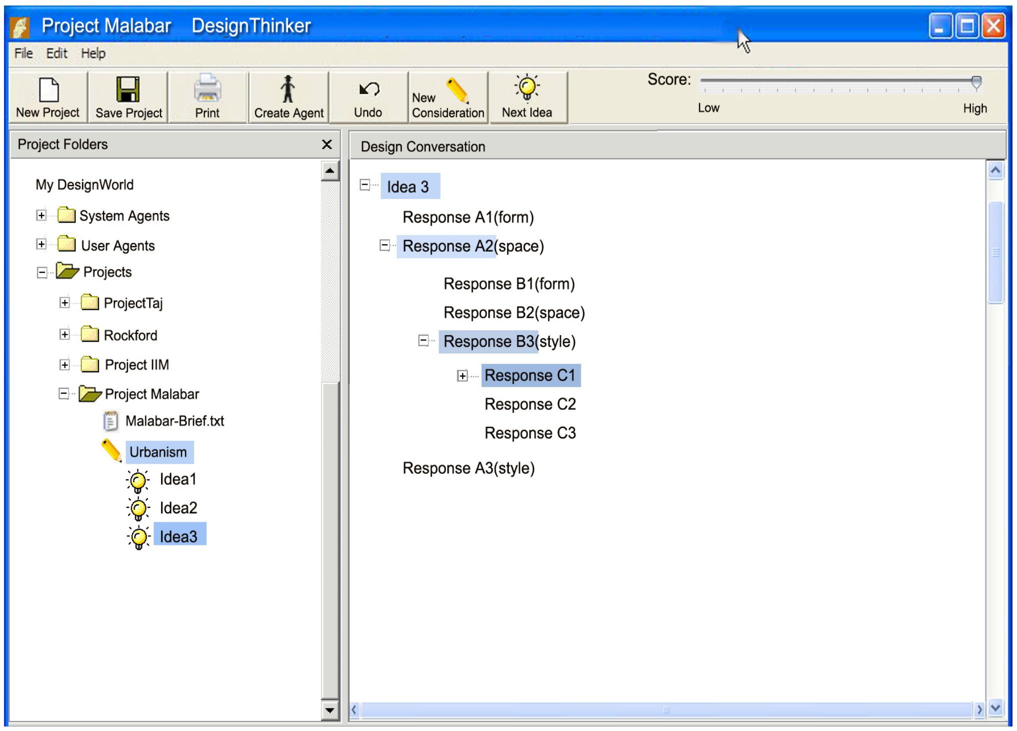


Figure 4: The Design Thinker as an 'Explorer' interface

Phase 2:

Once an *Idea* is selected by the user, it goes to the blackboard and is monitored by the domain agents. Based on the scoring methods and the scoring level, each domain agent returns one *Response* from its domain to the interface. These *Responses* have the respective agent's name displayed in brackets. The user can continue the conversation by selecting a response through a double click on the word. The selected response is highlighted on the interface and is transferred to the blackboard as a *Focus Term*. In return, the domain agents provide new *Responses* from their *Views* based on previous scoring methods and the conversation continues. On right-clicking a response, a user can choose to view the detailed definition and illustrations for the respective response to facilitate an explanation for the term.

In the course of the conversation, a user can choose to backtrack and follow a different thread of the conversation. This can happen in the following ways:

- The user can end the current thread of conversation and choose a previously unselected *Response* which adds a new dimension to the conversation.
- The user selects a new *Idea* for the *Design Consideration* and this follows a new course of conversation. The conversation for previous idea/s is stored under the respective ideas.

- The user can begin a new conversation by entering in a new *Design Consideration* for the current project.

Conclusion

The prototype 'Design Thinker' is currently in its final stages of implementation. Once complete, the prototype is to be tested by comparing a design session accomplished using the system to the traditional session. It is proposed that 2 testing groups shall be involved – a control group and a non-control group. An evaluation of the different measures of the design experience will be carried out as a group and on individual basis.

References

Goldschmidt, G. and Tatsa, D. (2005); How Good Are Good Ideas? Correlates Of Design Creativity; Design Studies*,* Vol. 26, No. 6 (pp. 593-611)

Haapasalo, H. (2000); Creative Computer Aided Architectural Design: An Internal Approach to the Design Process; Academic dissertation, Faculty of technology, University of Oulu, Oulu.

Heylighen, A., Neuckermans, H. and Bouwen, J.E. (1999); Walking on a Thin Line-Between Passive Knowledge and Active Knowing of Components and Concepts in Architectural Design; Design Studies*,* Vol. 20, No. 2 (pp. 211-235)

Jonson, B. (2005); Design Ideation: The Conceptual Sketch in the Digital Age; Design Studies*,* Vol. 26, No. 6 (pp 613-624)

Lawson, B. and Loke, S.M. (1997); Computers, Words and Pictures; Design Studies*,* Vol. 18, No. 2 (pp 171-183)

Lawson, B. R. (2001); The Context of Mind; Designing in Context; P. Lloyd and H. Christiaans; Delft, DUP Science (pp 133-148)

Markus, A.T. and Cameron, D. (2002); The Words Between the Spaces: Buildings and Language; London, New York; Routledge.

Medway, P. and Clark, B. (2003); Imagining the Building: Architectural Design as Semiotic Construction'; Design Studies*,* Vol. 24, No. 3 (pp 255-273)

Menezes, A. and Lawson, B. (2006); How Designers Perceive Sketches; Design Studies*,* Vol. 27, No. 5 (pp 571-585)

Oxman, R. (1996); Cognition and Design; Design Studies*,* Vol. 17, No. 1 (pp 337-340)

Schön, D. (1992); Designing as Reflective Conversation with the Materials of a Design Situation; Knowledge-Based Systems, Vol. 5, No. 1, (pp 3-14)

Schön, D. (1995); Reflective Practitioner: How Professionals Think in Action; New ed. England: Aldershot Press

Segers, N.M., de Vries, B. and Achten, H.H. (2005), Do Word Graphs Stimulate Design?; Design Studies*,* Vol. 26, No.6 (pp 625-647)

Suwa, M., Gero, J. and Purcell, T. (2000); Unexpected Discoveries and S-invention of Design Requirements: Important Vehicles for a Design Process; Design Studies*,* Vol. 21, No. 6 (pp 539-567)

Valkenburg, R. and Dorst, K. (1998); The Reflective Practice of Design Teams; Design Studies, Vol. 19, No. 3 (pp 249-271)

Enhancing the *Face* of Service-Oriented Capabilities

Kym J. Pohl
CDM Technologies Inc.
San Luis Obispo, California, USA
kpohl@cdmtech.com

Abstract

With today's focus toward discoverable web services, Service-Oriented Architectures (SOA) are becoming increasingly prevalent. To support an effective interaction between services and their clientele, the sophistication of the interface, or face, such services present is of critical importance. Without a rich, expressive nature, such services struggle to satisfy the industry promises of reuse, composability, and reduced inter-component dependencies. Especially relevant for domain-oriented applications, services must present sufficient levels of expression to allow for an effective exchange of relevant context. Further, such communication should be offered in an asynchronous manner to promote both work flow efficiency and limited coupling. This paper discusses several concepts and technologies that can significantly enhance the effectiveness of SOA-based capabilities. Technologies including JavaBeans, embedded property change management, and Object/Relational Mapping are leveraged to produce a client interface architecture rich in expressiveness, asynchronous efficiency, and industry standards.

Keywords

Service-Oriented Architecture (SOA), JavaBeans, property change management, object/relational mapping

Introduction

With the advent of web services (Antonion and Van Harmelen 2004, Daconta et al. 2003), the concept of a Service-Oriented Architecture (SOA) has received considerable attention (Erl 2005). Apart from offering benefits ranging from component reuse to runtime composition of capabilities, SOAs are laying the groundwork for dynamic semantic discovery. Considerably more powerful than technologies only able to convey a capability's structure (e.g., XML), the technologies associated with SOA endeavor to support the semantic discovery of a particular service's nature (i.e., domain of service, semantic expectations and implications). A critical element in attaining this goal is the sophistication of the interface services presented to their clients. This paper describes a combination of design concepts and technologies that can be exploited to produce the type of expressive client interfaces indicative of SOA.

JavaBeans Component Architecture and the *Property Change* Observation Model

The JavaBeans technology is one of the fundamental elements comprising the Java Component Architecture. JavaBeans, or simply *beans* for short, are essentially blueprinted objects exhibiting the following characteristics:

- Contain properties whose access is provided through standardized accessor methods (i.e., *getter* and *setter* methods).
- Serializable (useful for both persisting and streaming)
- May be enhanced with additional, application-specific methods for providing specific functionality to its users.
- Support *asynchronous* interaction via the firing of *property change* events(i.e., responses can be triggered by notification of the occurrence of previously *listened to* events)
- Can contain associated metadata (i.e., *BeanInfo*) offering a more precise description of its mechanics than can be provided through more simplistic mechanisms (i.e., Java Reflection)

Although a frequently employed technology in mainstream software development, the standardized protocols and embeddable event behavior promoted by the *beans* pattern provides a solid foundation upon which expressive, self-describing and notifying interfaces can be built. Indeed, it is the latter of these capabilities (i.e., property change management) that provides the asynchronous interaction model that enables the successful exploitation of parallel processing environments. As such, property change management warrants a brief discussion as to how this technology can be employed along with the distribution benefits it offers to SOA-based systems.

Property Change Management *(Observation Pattern)*

Complimenting the JavaBeans Component pattern, the property phange observation model establishes an implementable pattern allowing beans to essentially *observe* changes in the *bound* properties of other beans. Observing objects are required to implement specialized interfaces that are automatically invoked whenever the particular condition occurs. Although fundamentally scoped to changes in individual properties, this pattern can be extended to support multi-faceted events occurring across heterogeneous sets of beans. Further, although this mechanism operates locally within a single Java Virtual Machine (VM), such functionality can be extended to seamlessly operate across any number of networked VMs through incorporation of an appropriate transport mechanism (e.g., JMS, CORBA, JDO, etc.) together with a degree of distribution management logic. Regardless of whether operating on a purely local basis or operating in a distributed fashion across a network, users of such *publish/subscribe* facilities operate against the exact same interface, transparent to whether this facility was housed locally or across the network.

The JavaBeans component pattern combined with property change management has become an industry standard. Together, these two technologies are an effective mechanism supporting the asynchronous, event-driven interaction model inherent in parallel processing paradigms (i.e., clients are free to perform other tasks while their requests are being processed). Further, this asynchronous model assists in promoting loosely coupled architectures where component interactions occur as initiations of events paired with any number of anonymous reactions. It should be noted that the latter of these features also aligns well with the loosely coupled and extensible architectures of Aspect-Oriented programming.

Domain-Centricity

Some capabilities are inherently domain-centric. That is, a domain-centric capability is one that operates over an expressive model closely representing concepts and entities conceivable in some reality. Such subject matter may be tangible or intangible, represent actual reality or some hypothetical variation. Regardless, however, such descriptions comprise the domain over which the capability operates. For example, a capability for planning the delivery of goods will quite plausibly operate over certain notions fundamental to the domain of logistics support (e.g., requirements, transports, delivery routes, goods, time schedules, known impediments, etc.) In essence, these notions form the subject matter upon which the capability operates.

In cases where interaction with a capability's clientele makes significant references to such notions and entities, it is useful to structure the client interface around expressive, domain-specific object models, also known as ontologies (Figure 1). Defining an interface explicitly in terms of the objectified *context* that is conveyed between parties offers both an expressive and more natural language by which services and clients can interact. Such enriched discourse can be substantially more effective and natural than the more traditional approach where such domain-specific context is parameterized into a limited set of invocable functions. Further, defining an interface in terms of such objectified, domain-specific context promotes the efficiency and decoupling offered by asynchronous, event-based interaction as well as the architectural simplicity and elegance incurred with an interaction model comprised of basic, object-level operations (i.e., object creation, manipulation, etc.). Further, combining this concept of domain centricity with the complimentary *bean* and *observer* patterns described above yields a standardized, yet expressive, client interface that supports a decoupled, asynchronous interaction model.

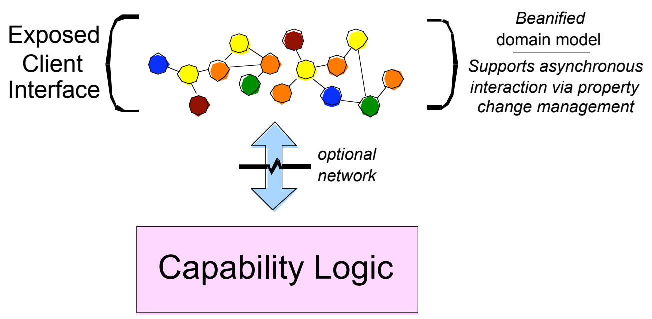


Figure 1: Domain-Centric Client Interface

To illustrate how these complimentary patterns function together, consider the *Delivery* service briefly described above. The client interface offered by such a service could be composed of an expressive domain model that includes explicit object –level descriptions representing *Requirements*, *Constraints*, *Resources*, etc. Populated by the requesting client, this inter-related cluster of ontology objects can be used to effectively convey the particular problem definition the service is being engaged to solve. Creation and population of instances of these objects would, in turn, trigger the *observing* Delivery service to analyze this information in conjunction with its knowledge of the environment within which the solution would execute (i.e., weather, traffic, security risk, etc.) the deliveries are to be executed within. Because the interaction model is event-driven, the requesting client is free to perform other tasks as its request is being processed. Upon formulating a suitable solution, the service would follow the same asynchronous interaction model as before producing a populated ontology fragment describing the proposed solution (e.g., delivery trips comprised of aspects including timing, stowing, sequencing, routing, mitigation instructions for possible impediments, etc.) To receive such results, clients would simply *observe* the creation/modification of such model fragments relating to their original problem context.

The example above illustrates the use of the property change observation mechanism of the JavaBeans Component Architecture, including the *asynchronous* interaction model that it promotes. The example also highlights the structuring of a client interface around the domain-specific notions that are the fundamental basis of a contextual discourse. Further, such domain-centric interface also provides an effective means of decoupling clients from a capability's underlying architecture (i.e., functional libraries, information management infrastructure, middleware used to distribute the service across a network, etc.).

Objectifying Relational Schemas

The client interface architecture described above can also be applied to capabilities whose domain models are internally housed within relational environments (e.g., RDBMS, etc.). Whether fully formed services, or simply managed sources of content (i.e., database), such components can expose their domains equally well as collections of interrelated domain objects

(Figure 2). Such objectification can be achieved through the application of any number of Object/Relational Mapping (O/RM) technologies (i.e., Hibernate, TopLink, EJB3, JDO, etc.) aimed at addressing the object/relational impedance mismatch (Hay 1996). These technologies offer the ability to expose a relational model in an object-oriented form, complete with support for potentially extensive mappings between object fields and corresponding table columns.

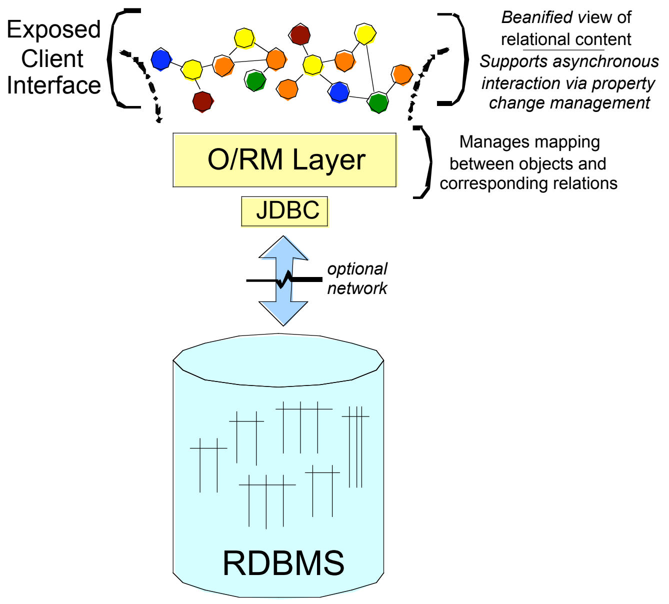

Exposed Client Interface
Beanified view of relational content
Supports asynchronous interaction via property change management
O/RM Layer
Manages mapping between objects and corresponding relations
JDBC
optional network
RDBMS

Figure 2: Presenting an Objectified View of Relational Content

A key part of O/RM technologies is the specification (typically in the form of *metadata*) and subsequent runtime management of mappings that effectively tie both diverse worlds together (i.e., object classes and relational tables). Finally, by applying the complimentary JavaBeans and property change management technologies, clients to such relational environments can be presented with domain-centric interfaces supporting an efficient, asynchronous interaction model. As such, relationally-oriented capabilities can also enjoy the contextual, efficiency, and decoupling benefits afforded by presenting clients with a domain-centric, event-driven interface.

Conclusion

With the increased application of Service-Oriented Architecture, there is a growing need to address the ease and efficiency by which such services are employed. This need is even greater when domain-centric capabilities are considered. Technologies that promote standardized, self-descriptive, expressive, and efficient interaction are paramount in supporting the collaboration-intense nature of an evolving, domain-centric and dynamically discovering semantic web topology (Antonion and Van Harmelen 2004, Daconta et al. 2003).

References

Antoniou G and F. Van Harmelen (2004), “A Semantic Web Primer”, MIT Press, Cambridge, Massachusetts.

Daconta M., L. Obrst and K. Smith (2003), “The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management”, Wiley, Indianapolis, IN.

Erl T. (2005), “Service-Oriented Architecture (SOA: Concepts, Technology, and Design”, Prentice Hall Service-Oriented Computing Series, Prentice Hall, Englewood, NJ.

Fowler, M. (1997), “Analysis Patterns: Reusable Object Models”, Addison-Wesley, Reading, Massachusetts.

Fowler M., D. Rice, M. Foemmel, E. Hieatt, R. Mee, and R. Stafford (2003), “Patterns of Enterprise Application Architecture”, Addison-Wesley, Reading, Massachusetts.

Hay D. (1996), “Data Modeling Patterns: Conventions of Thought”, Doset House Publishing, New York, NY.

Karsai G. (2000), “Design Tool Integration: An Exercise in Semantic Interoperability”, Proceedings of the IEEE Engineering of Computer Based Systems, Edinburgh, UK, March.

Pohl J. (2001), “Information-Centric Decision-Support Systems: A Blueprint for Interoperability”, Office of Naval Research (ONR) Workshop hosted by the CAD Research Center in Quantico, VA, June 5-7.

Pohl J, A Chapman, K Pohl, J Primrose and A Wozniak 1997), “Decision-Support Systems: Notions, Prototypes, and In-Use Applications”, Technical Report, CADRU-11-97, CAD Research Center, Design Institute, College of Architecture and Environmental Design, Cal Poly, San Luis Obispo, CA, January.

Rodrigues L. (1998), “The Awesome Power Of Java Beans”, Manning Publication, Greenwich, CT.

Alternative Paths to Intelligent Systems

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 examines the three prevalent approaches to Artificial Intelligence (AI), namely symbolic reasoning systems, connectionist systems, and emergent systems based on the principles of the subsumption theory. Distinguished by their top-down and bottom-up mechanisms all three approaches have strengths and weaknesses. While the logical reasoning approach is precise and well supported by mathematical theories and procedures, it is constrained by a largely predefined representational model. Connectionist systems, on the other hand, are able to recognize patterns even if these patterns are only similar and not identical to the patterns that they have been trained to recognize, but they have no understanding of the meaning of those patterns. The subsumption approach appears to overcome many of the weaknesses of the other two approaches in theory, but there is concern that it may not scale to more complex real world applications.

The author points out that in addition there are weaknesses that all three AI approaches share, namely inability to deal with exceptions, lack of mechanisms for analogous comparisons, and very primitive conceptualization capabilities at best. It is noted that the human agent performs decidedly better in these areas.

The paper concludes with the proposition that only a hybrid approach holds sufficient promise to meet the full expectations of intelligent systems. It is further suggested that this hybrid approach should include the contributions of the human agent as an integral component of the intelligent system, in most cases.

Keywords

Actuator, agents, Artificial Intelligence (AI), connectionist systems, context, data, embodied, emergent, information, intelligence, neural network, neurode, ontology, representation, sensor, situated, subsumption, symbolic reasoning, synapse.

Computation and Data-Processing

The need for devices with computational capabilities that exceed manual processes by several orders of magnitude, in terms of speed, was driven largely by mathematicians and physicists. During the first half of the 20th Century it was not uncommon for persons with doctorate degrees to spend weeks on the tedious solution of large sets of simultaneous equations for solving partial differential equations. These mathematical solutions were required for the preparation of tables that served as essential practical aids for many military and navigational purposes.

In this respect even the first, by today's standards, slow and clumsy electronic machines did not disappoint their creators. The ENIAC (Electronic Numerical Integrator And Computer) that was completed in 1946 at the University of Pennsylvania under the guidance of John Mauchly and J. Presper Eckert was able to perform a set of calculations that would have taken conventional calculating machines 40 hours, in 20 seconds. Even though it operated at only 100,000 pulses per second[1], this represented a hundred fold increase in calculation speed over the conventional electro-mechanical calculator technology and a more than one thousand fold increase in calculation speed over manual calculations (Goldstine and Goldstine 1982, Rojas and Hashagen 2000).

However, it soon became apparent that apart from computation there was another growing manual task that was in need of assistance, namely data-processing. Much of the data came in the form of textual data items that needed to be stored, sorted, and analyzed.

In more recent years, with advances in data storage technology accompanied by considerably decreasing costs, there has been an enormous increase in the amount of data stored and processed by computers. This has led to a bottleneck because while computers are able to store and process data as rapidly as they are able to compute numbers, they are unable to interpret the meaning of the data being processed. This essential task was left to the human users, because only they understood the context within which the data was being generated and destined to be used (Figure 1).

With rising expectations that the ability to store and process ever greater amounts of data should lead to better quality and faster planning and decision-making capabilities, the human user increasingly fell behind. This has become particularly apparent in the military intelligence and homeland security communities. High visibility examples include the World Trade Center catastrophe of 11 September 2001. As soon as the initial emergencies had been more or less dealt with, both the general public and the press media asked the obvious questions: Why wasn't this tragedy prevented? Were there not signs that the attack was about to take place? How could the intelligence agencies have been unaware that something sinister was being planned? The almost immediate truthful response was: *There is so much data out there that our intelligence analysts are often unable to see the forest for the trees*. In other words, the intelligence analysts were simply overwhelmed by the continuous flow of data through the many connected and also unconnected intelligence networks.

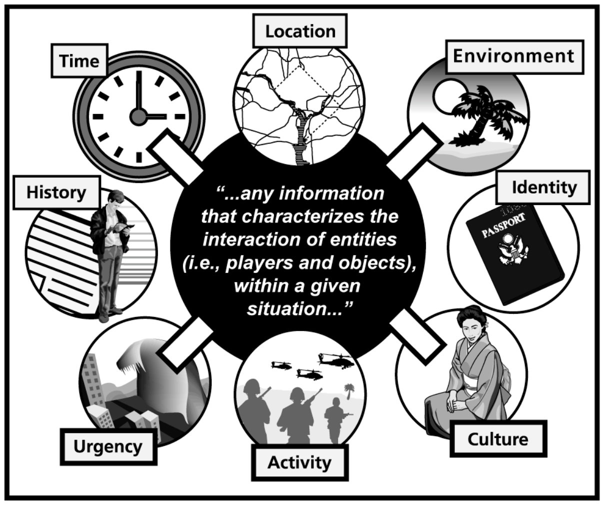
Based on human expectations and an increasing demand for *actionable information*[2] the pressure is mounting for computers to be able to not only store and manage data, but also to be able to interpret the meaning of the data. In particular, to be able to automatically detect and interpret the changes in data that occur as a direct result of events in the knowledge domain to which the data pertains.

So how can an electronic computing device automatically reason about data, recognize patterns, and perform any of the tasks that we normally associate only with human intelligence? This is a question that has been a subject of intense study by small groups of researchers for the past 50

[1] This is equivalent to a clock speed of 0.1 MHz, compared to a fairly standard laptop computer today (2007) operating at over 2 GHz (i.e., 2,000 MHz).

[2] The term *actionable information* has been coined by the military to describe data that have been interpreted, filtered, and are ready to form the basis of human decision-making sequences leading to actions.

years. Initially this relatively small Artificial Intelligence (AI) community of researchers had high hopes of being able to produce systems that could match human intelligence in the short period of time of a few years. It turned out that these expectations were far too optimistic. The disappointment was so great that for some time AI research fell into disrepute. Funding sources dried up and research in this field was considered by many to be unproductive and ill advised. This over-reaction was unfortunate, because it is now generally recognized not only that there is a real need for intelligent systems but also that such systems will eventually become reality. In fact, it is even argued by some that there is a gradual merging of biology and technology, a convergence that is an essential part of the intellectual evolution of the human species (Kurzweil 1999).


Location
Time
Environment
History
Identity
"...any information that characterizes the interaction of entities (i.e., players and objects), within a given situation..."
PASSPORT
Urgency
Activity
Culture
Figure 1: Typical components of context

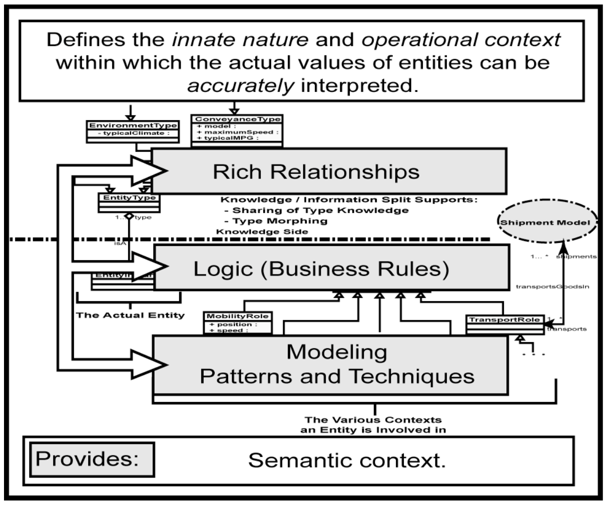


Figure 2: Ontology representation

Artificial Intelligence Approaches

Fundamentally, there have been two principal approaches to the development of intelligent systems, generally referred to as the *top-down* and the *bottom-up* approaches. In the top-down approach researchers have tried to emulate the rationalistic reasoning capabilities of the human cognitive system. In other words, these efforts have been focused on constructing virtual representations of real world situations and conditions in computer software. The strategy is that these virtual information models will then provide some measure of context to software modules with built-in reasoning engines.

The bottom-up approach has taken an entirely different path. The proponents of this approach have attempted to simulate the biological basis of the human nervous system. Referred to as the *connectionists*, they set out to construct mathematical engines to simulate the neurons and synapses that are an integral component of all forms of intelligent life. In this way they were able to achieve some measure of pattern recognition and matching that appears to be the foundation of human intelligence.

Symbolic Logic Systems

This top-down approach recognizes that *context* is a most likely prerequisite for all rational thought processes such as: the interpretation and filtering of data; the recognition of useful and actionable information; and, the problem solving processes that are used in planning, evaluation, and decision-making endeavors. Accordingly, the proponents of this approach established conventions and languages such as the Unified Modeling Language (UML) and the Object Constraint Language (OCL) to form the necessary building blocks for constructing complex virtual models of real world knowledge domains (Booch et al. 1997, Larman 1998).

Initially, these representational models were referred to as object models because they defined real world entities as objects with attributes and behavioral characteristics. The objects were then associated with each other to simulate the relationships that allow human reasoning to determine the meaning and significance of changes in data that result from events in the real world. As these representational models and the methods for their construction became more sophisticated an increasing number of the reasoning capabilities could be moved from the reasoning engines (i.e., agents) into the representation (i.e., the virtual model). Today, these virtual models are more aptly referred to as ontologies because they do, true to their dictionary definition, attempt to represent all of the concepts and notions in a particular knowledge domain.

An ontology is an information model, rich in relationships, which describes the context of a real world situation or problem domain such as the management of goods movement across international borders, or the design and manufacturing process of an engineering product, or the command and control decision-making environment of a military battlespace (Figure 2). It is not limited to the modeling of physical entities such as buildings, roads, persons, activities, climatic factors, weapons, and vehicles. An ontology can also model abstract concepts such as threat, privacy, consumability, mobility, stature, and time. Some of these entities are of course easier to model than others. For example, the behavioral characteristics of an aircraft can be quite easily described in terms of attributes such as cruising speed, range, dimensions, maximum payload, and so on. Even the relationships between an aircraft and its landing and take-off requirements, its refueling and crew needs, and its maintenance schedule can be comprehensively modeled with a high degree of accuracy. However, to build a comprehensive model of the cultural characteristics of a nation or even an organization is a much more difficult undertaking. The reason for this difficulty is not necessarily due to the shortcomings of the available representational tools and methodologies, but rather the inadequacies of our own human understanding of these human behavioral characteristics.

The reasoning engines or software agents that are designed to navigate these somewhat simplistic virtual models are commonly in the form of *if-then* rules (i.e., if certain conditions or antecedents are reflected in the current state of the model then certain inferences are automatically made). These inferences are normally expressed as actions that are immediately executed by the agent. In this way software agents are able to autonomously communicate with other software agents or human users, monitor events by tracking data changes, retrieve data from external sources, request and provide specialized services, pursue interests and objectives, and accomplish at least low level learning tasks.

While this may seem impressive, and is certainly orders of magnitude more powerful than rote data-processing without context, it is still quite limited in comparison with human cognitive capabilities. This becomes particularly apparent when we consider human capabilities in the areas of intuition and common sense.

More specifically, difficulties with this top-down approach are routinely encountered in several areas. It would appear that the entire mechanism of antecedents (conditions) and conclusions (actions) is overly simplistic. Often there are too few matching conditions to unambiguously determine which rule most clearly fits a particular situation. The ability to create during execution a new rule that combines those elements of two or more rules that fit the situation certainly exists in theory but is difficult to implement effectively in practice. On the other hand, there are often too many matching conditions. The available recourses in this case are by no means foolproof. Some of the more common alternatives include the implementation of a priority scheme to select one rule, or invoking some other mechanism such as case-based reasoning to select the most appropriate rule, or to create a new rule that combines the most desirable features of all the rules that fit the situation, or perhaps it may be more useful to pursue multiple alternative paths in parallel.

In summary, the advantages of the top-town approach are basically threefold. First, the formal symbolic logic on which this approach is based provides clarity and verifiable precision. Second, there is the availability of well established mathematical theories and procedures, and third, the similarity to the human reasoning process is obviously very attractive to us human beings.

However, on the downside there are also some serious disadvantages that apply to this AI approach. Foremost among these is the intrinsically static nature of the representation model. It forces the top-down approach to adhere to a largely predefined and explicit representation of objects, the behavioral characteristics of those objects, and the relationships among the objects. Further advances in methods that will allow ontologies to be extended, shared, and merged during execution are urgently needed and likely to become available in the foreseeable future. While the availability of such methods will greatly increase the power of ontology-based systems, there will remain the constraints imposed by a predefined and strictly ordered view of a world that in reality is subject to continuous change due to the interactions of the elements in its domain.

Also, due to its reliance on an explicit representation the top-down approach cannot easily deal with exceptions (Minsky 1990). Yet, in the real world we are often reminded of the importance of the exceptions to the general rules. This is perhaps an unfair criticism of the top-down approach because we really have not devised any formal mechanisms for dealing with exceptions. Statistical methods were developed to establish norms and deviations from these norms. Fuzzy logic provides a set of mathematical methods for establishing the certainty of inferences. However there is no equivalent mathematical method available for identifying patterns derived from the confluence of exceptions.

Another drawback of the top-down approach is its apparent inability to support the formulation of analogies. An example of such an analogy capability is the ability to conduct conceptual searches in a distributed database management system (DBMS). A traditional DBMS typically supports only factual searches. In other words, users and applications must be able to define precisely and without ambiguity what data they require. In complex problem situations users rarely know exactly what information they require. Often they can define in only conceptual terms the kind of information that they are seeking. Also, they would like to be able to rely on the DBMS to automatically broaden the search with a view to *discovering* information. This suggests a need for the ability to formulate search strategies based on incomplete definitions. It should be possible to infer from rather vague information requests and knowledge of the problem context, a set of executable query procedures that will lead to the discovery of analogous information domains.

Finally, formal information models are unable to represent the wealth of information and knowledge that allows us human beings to exercise common sense. This became a serious criticism of early AI research and still remains today one of its greater weaknesses.

Connectionist Systems

The objective of the bottom-up connectionist approach is to emulate the biological basis of the human nervous system. The human brain contains over 100 billion computing elements (i.e., neurons), which communicate throughout the human body through nerve fibers with over 100 trillion interconnections (i.e., synapses). Some neurons have only a few synapses and others may have thousands. This network of neurons is responsible for all of the human phenomena that are referred to as thought, memory, emotion, and cognition.

The principal capability of the human brain appears to be related to the recognition and processing of patterns. This human pattern matching capability applies to speech communication, the recognition of persons and objects, the performance of tasks, and reasoning. The connectionist approach to emulating this pattern matching capability depends on the construction of a network of interconnected nodes that are capable of sending signals to each other. The strength or numerical values of these signals are based on the cumulative application of mathematical weighting functions.

The nodes can be thought of as artificial neurons and are referred to as *neurodes*. Each neurode is essentially implemented as a mathematical *transformation function*[3] that associates a given level of input signals with a particular level of output. The neurodes are generally connected in multiple layers that include an input layer, one or more intermediate or *hidden* layers, and an output layer. Each input neurode is connected to each neurode in the next layer. These connections or artificial synapses can be unidirectional or bidirectional. Typically, due to the many interconnections, each neurode receives a large number of input signals. These input signals are accumulated by the neurode until they exceed a threshold value, at which time the neurode will send an output signal to other connected neurodes (Figure 3).

When implemented in software executing on a digital computer, each input layer neurode receives a starting value (signal) between 0.0 and 1.0. If the cumulative input values exceed a predefined threshold value then the neurode will *fire* and send identical output values (signals) to each second-layer neurode. Each of these second-layer neurodes will multiply the value (signal) received by a weighting factor. These values are summed and as soon as the combined value exceeds the threshold value the neurode will *fire*, and so on.

The advantages of the connectionist approach are very different from the advantages of the top-down approach and intuitively attractive to our human viewpoint. Apart from their apparently

[3] The *fractional Fourier transform* (FRFT) is a linear transformation that is often used in pattern matching neural networks.

elegant mathematical formulation, neural networks perform quite well even with incomplete input and can recognize conditions that are similar but not identical. Moreover, any particular neural network can normally be trained to recognize several kinds of patterns. For example, the same neural network can be trained to recognize most (if not all) letters of the alphabet. Neural networks have been successfully applied to perform quite complex pattern matching tasks, such as navigating a car on a road with other traffic at speeds above 50 mph.

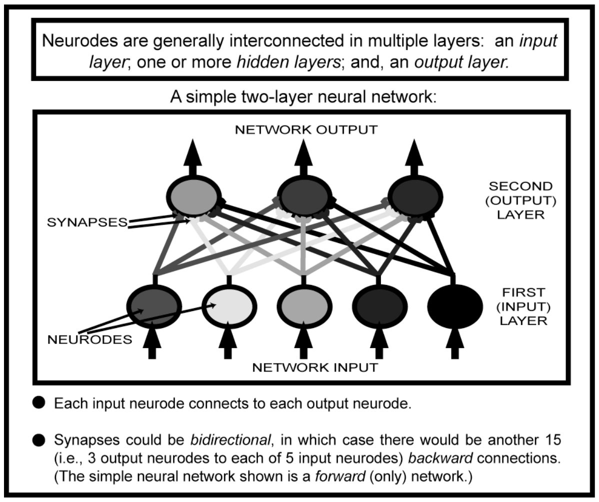

Neurodes are generally interconnected in multiple layers: an *input layer*, one or more *hidden layers*; and, an *output layer.*
A simple two-layer neural network:
NETWORK OUTPUT
SECOND (OUTPUT) LAYER
SYNAPSES
FIRST (INPUT) LAYER
NEURODES
NETWORK INPUT
● Each input neurode connects to each output neurode.
● Synapses could be *bidirectional*, in which case there would be another 15 (i.e., 3 output neurodes to each of 5 input neurodes) *backward* connections. (The simple neural network shown is a *forward* (only) network.)

Figure 3: Neural network components

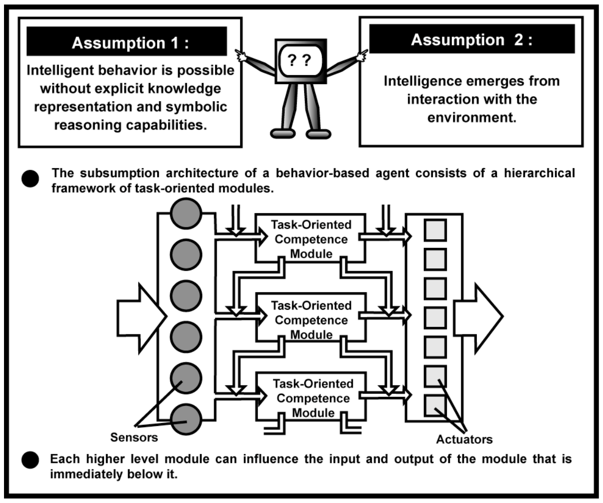

Assumption 1 :
Intelligent behavior is possible without explicit knowledge representation and symbolic reasoning capabilities.
? ?
Assumption 2 :
Intelligence emerges from interaction with the environment.
● The subsumption architecture of a behavior-based agent consists of a hierarchical framework of task-oriented modules.
Task-Oriented Competence Module
Task-Oriented Competence Module
Task-Oriented Competence Module
Sensors
Actuators
● Each higher level module can influence the input and output of the module that is immediately below it.

Figure 4: Subsumption architecture

However, the connectionist approach also has some intrinsic disadvantages. First and foremost, there is absolutely no *understanding* within the network of the meaning of the pattern that has been recognized. All that the neural network has been able to achieve is to generate a set of very similar output values (signals) every time it receives a different but also similar set of input signals. For example, if a neural network has been trained to recognize the alphabetic character *H* then it will recognize this letter with reliability only if it is presented to it in the same surroundings. Let us assume that the letter *H* was located at the bottom left side of a computer screen when the network was trained. Then there is no guarantee that the network will recognize the same letter *H* if it is moved to the top right side of the screen. In fact, if there are other competing images around the *H* in one or both locations it is likely that the network will fail to recognize the *H* pattern in the new location.

The above example suggests quite correctly that there is in fact little theoretical understanding of exactly how the mathematical representation leads to the pattern matching capability of the neural network. Attempts to gain such an understanding are confounded by the fact that the knowledge within the internal nodes of the hidden layers is not readily accessible. Also, the weighting coefficients that modify the input values (signals) in each node cannot be changed like software can change the content of the memory cells in a digital computer in the top-down approach.

Finally, it is of course very difficult to explore alternatives with neural networks. There is no reasoning capability because the network simply matches patterns. It has no understanding of the semantics of those patterns and is therefore unable to build on its initial recognition achievement with higher level tasks. Of course such tasks could be performed by external top-down systems that map the output of the neural network to a symbolic representation for reasoning purposes.

Emergent Systems

A bottom-up approach that is quite different from the connectionist methodology was first proposed in the 1980s by Rodney Brooks, who has been leading an AI research effort in the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology (MIT) for the past 30 years. Brooks argues that the top-down approach is fundamentally flawed for at least three reasons (Brooks 1991). First, most of the activities performed by humans on a daily basis are routine and do not involve problem-solving or planning[4]. Second, complete models of real world environments are impossible to construct because they are subject to change. Intelligence can emerge from subcomponents interacting with each other and the environment through sensory mechanisms. Accordingly, Brooks has been conducting most of his research with mobile robots that are able to interact with the environment in which they exist. Third, agents can have beliefs and goals without actively reasoning about high level semantics. This does not necessarily imply that robots should not reason about their environment, but rather that the semantic representation that is required for reasoning will need to be built by the robot during its interactions with the environment.

Brooks first introduced the notions of the *subsumption* theory in 1986 (Brooks 1986). The term *subsumption* derives from the tight coupling between emergent intelligence and the real world environment. The subsumption architecture of a behavior-based agent (robot) consists of a hierarchical framework of task-oriented modules (Figure 4). Taking input from sensors and providing output to actuators, each higher level module can influence the input and output of the module that is immediately below it. Triggered by its sensors a robot agent dynamically builds a temporal model of the real world that surrounds it. Objects and relationships are primarily relevant as they are sensed and only secondarily important within the larger context of a more complete model of the world (i.e., the kind of model that is a fundamental prerequisite of the top-down approach). The subsumption theory is based on four major pillars, namely that robot agents are situated, embodied, intelligent, and emergent.

A robot agent is *situated* because it continuously refers to its sensors rather than an internal model of the world. Responding quickly to its sensor inputs, the robot is forced to build a temporal model of its surroundings relative to itself rather than an external framework. For example, the robot would refer to a sofa as the obstacle that is right now to its right, rather than *object 7*, which is a sofa. In other words, the robot agent is required to learn about its environment by interpreting its real world experiences with little (if any) initialized knowledge. Clearly, this approach is particularly appropriate in a dynamically changing world in which the past state of the world provides little reliable information about the current and future states.

The notion of *embodiment* is based on the fact that the physical presence of the robot agent forces it to potentially deal with all issues that its sensors and actuators are capable of processing. The assumption is that only an *embodied* agent can be validated in terms of its autonomous capabilities and its intelligence. Therefore, timely perception and action in preference to strategic planning and problem-solving are likely to be the most challenging behavioral capabilities of

[4] This view is shared by Jeff Hawkins whose Hierarchical Temporal Memory (HTM) theory is based on his research of the functions of the neocortex (Hawkins and Blakeslee 2004).

such an agent. However, this also suggests that some degree of redundancy will be necessary because such agents are likely to be vulnerable to sensor malfunction.

Brooks argues that robot *intelligence*, like human intelligence, is largely a function of the degree of complexity of the environment rather than its own internal complexity. Since intelligence is determined by the dynamics of interaction with the environment, it is often difficult and not necessarily useful to draw a distinction between intelligence and successful environmental interaction. Therefore, similar to human evolution where the development of perception and mobility capabilities took much longer to develop than reasoning capabilities, the intelligence of a robot agent depends more on its dynamic interaction capabilities than its reasoning capabilities.

Finally, the intelligence of a robot agent *emerges* through the interaction of components. These components are best focused on behavior producing tasks than functional information processing tasks. Accordingly, the components of these behavior-based robots are designed to produce the required environmental interaction and mobility capabilities collectively. It is therefore difficult to identify the seat of intelligence of a robot agent because the intelligence is the result of the interaction of many contributing capabilities. In other words, higher level intelligent behavior *emerges* from lower level behavioral capabilities through a process of repetitive learning.

In summary, subsumption or behavior-based systems are reactive rather than proactive systems, whose planning interests and capabilities are driven largely by unexpected needs. Their strength lies in the fact that without resorting to any central symbolic representation they are capable of: making predictions and forming expectations about their environment; developing plans that relate to their immediate needs; and, formulating and implementing goals.

At the same time, it is reasonable to question whether the bottom-up subsumption systems will be able to scale to more and more complex real world environments. This will most likely depend on the ability of such robot agents to learn from their interactions with the environment in which they operate. Will they be able to respond with sufficient speed to their sensor inputs to progressively develop a level of intelligence that is several orders more sophisticated than their foundational sense and response mechanism? Will they be able to build an experience-based pattern identification and problem-solving capability?

The Path Ahead

Clearly all three of the AI approaches are facing major obstacles. The symbolic reasoning approach depends on a largely predefined virtual model of the real world. While some aspects of the real world environment are fairly static, there are other features that are subject to continuous change as the players in the real world interact with each other and their environment. For example, the furniture in virtual any building space will be moved around due to the various interactions of the building occupants. Even more apparent chaos will exist at certain times in a manufacturing plant or supermarket. Since these changes cannot be adequately predefined there is a need for the virtual model to adapt as the real world environment changes. The ability to extend ontologies during execution is absolutely necessary and will certainly be helpful, however, the first sign that something has changed in the environment will likely come from the agents in the environment (i.e., from the bottom up). Therefore, there is a need for these agents to be able to identify and adapt to the changes before the virtual representation model can be corrected from the top down.

The connectionist systems can recognize patterns but have no understanding of what they have recognized. In other words, they can detect non-literal similarities but need some other mechanism to determine what action (if any) should be taken in response to the detected situation. The strengths and weaknesses of the symbolic reasoning and connectionist systems appear to be somewhat complementary. For example, whereas logical reasoning systems are very limited in their ability to detect non-literal similarities, neural networks can recognize conditions that are similar but by no means identical. Also, if a neural network recognizes a pattern then a symbolic reasoning system working in conjunction with the network may be able to determine the meaning and significance of the condition within the context of its virtual representation model.

The subsumption approach embodies several very powerful notions that overcome some of the theoretical objections to both the symbolic reasoning and connectionist approaches. Its key constructs of *situated*, *embodied*, and *emergent* are intuitively obvious. Also, the configuration of sensors, task-oriented competence modules, and actuators in the subsumption architecture is seductively simple in theory. However, can it be implemented in practice in systems that are useful beyond mere demonstrations or toys? In other words, is this theoretically very promising approach scalable in practice?

There are other serious shortcomings that apply to all three AI approaches. First, there remains the problem of dealing with exceptions. It would appear that the representational models of the top-down approach must by their very nature always be inclusionary in character by adhering to the principles of consistency, regularity, and conformity. The subsumption approach perhaps comes closest to dealing with exceptions because the agent robots do not necessarily draw a distinction between norm and exception until they start to overly rely on their experience. Second, there do not appear to be any effective mechanisms for dealing with analogous comparisons in computers, and third, only very primitive conceptualization capabilities such as case-based reasoning have been demonstrated in symbolic systems to date.

So, what is the path ahead? It does not appear that any one approach is sufficiently strong to meet the full expectations of intelligent systems. A hybrid approach is likely to be necessary. However, it would be well to consider the human user not only as a necessary but also as a beneficial contributor in most intelligent system environments. If we consider technology as an enabler of human capabilities then it would seem appropriate that the strengths of the human should form an integral part of the intelligent environment. In the same way that at least some of the strengths and weaknesses of the top-down and bottom-up AI approaches complement each other, the human strengths in the areas of analogous thought and conceptualization complement the weaknesses of all three AI approaches.

References

Booch G. J. Rumbaugh, and I. Jacobson (1997); 'The Unified Modeling Language (UML) User Guide'; Addison-Wesley, Reading Massachusetts.

Brooks R. (1991); 'Intelligence Without Representation'; Artificial Intelligence, Volume 47, (pp.139-159).

Brooks R. (1991); 'Intelligence Without Reason'; A.I. Memo No. 1293, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, Cambridge, Massachusetts.

Brooks R. (1990); ‘Elephants Don’t Play Chess’; Robotics and Autonomous Systems, Volume 6, (pp 3-15).

Brooks R. (1986); ‘A Robust Layered Control System for a Mobile Robot’; IEEE Journal of Robotics and Automation, RA-2, April, (pp. 14-23).

Goldstine H. and A. Goldstine (1982); ‘The Electronic Numerical Integrator and Computer’; in *The Origins of Digital Computers: Selected Papers*, Springer-Verlag, New York, New York (pp. 359-373).

Hawkins J. and S. Blakeslee (2004); ‘On Intelligence’; Times Books, Henry Holt and Company, New York, New York.

Kurzweil R. (1999); ‘The Age of Spiritual Machines: When Computers Exceed Human Intelligence’; Viking, Penguin Group, New York, New York.

Larman C. (1998); ‘Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design’; Prentice Hall, Upper Saddle River, New Jersey.

Minsky M. (1990); ‘Logical vs. Analogical or Symbolic vs. Connectionist or Neat vs. Scruffy’; in Winston P. *Artificial Intelligence at MIT: Expanding Frontiers*, Volume 1, MIT Press, Cambridge, Massachusetts.

Minsky M. (1982); ‘Why People Think Computers Can’t’; AI Magazine, Volume 3, No. 4, Fall.

Rojas R. and U. Hashagen (2000); ‘The First Computers: History and Architecture’; MIT Press, Cambridge, Massachusetts.