InterSymp – 2004

16th International Conference on Systems Research, Informatics and Cybernetics

(July 29-August 5, 2004, Baden-Baden, Germany)

Pre-Conference Proceedings of the Focus Symposium on

Intelligent Software Systems for the New Infostructure

Friday, July 30, 2004

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: 1-894613-41-4

Preface

Over the past several years the papers in this annual series of symposia have increasingly centered on the realization of a human-computer collaboration environment in which computer-based software agents with reasoning capabilities provide meaningful support to human decision makers. It is therefore quite appropriate that the first paper in the 2004 Proceedings should address the historical evolutionary path of 'intelligent' software leading to the goal of a semantic Web environment. The realization of this goal is now in sight, driven by public security threats that are increasingly relying on technology for effective countermeasures.

In the U.S. during the past two years there has been increasing government activity focused on the creation of a communication environment that will provide seamless horizontal and vertical connectivity among all echelons in support of an effectively coordinated disaster response capability. The new U.S. Department of Homeland Security (DHS) has coined the word ***Infostructure*** to describe this environment. For the U.S. Department of Defense (DoD) the ***Global Information Grid*** or ***GIG*** is the vision that will guide the implementation of such a capability, in the form of an integrated network of knowledge management services. Succinctly stated both the Infostructure and the GIG are envisioned as a net-centric environment of seamlessly interconnected data sources and utilization capabilities. This includes "… the globally interconnected, end-to-end set of information capabilities, associated processes, and personnel for collecting, processing, storing, disseminating, and managing information on demand to warfighters, defense policymakers, and support personnel." (Stenbit 2003).

These parallel and closely related U.S. efforts (i.e., the Infostructure-GIG vision) are driven by the increasing need to automate at least the lower level data interpretation tasks that have been the almost exclusive province of the human users of computer systems for the past 40 years. This has become necessary for several reasons. First, the increased ability to collect and store data in computers has created a bottleneck. The need to interpret the vast amounts of data has simply exceeded the availability of human resources. Second, human resources are desperately needed for higher-level information analysis to counteract increasing threats from adversaries. Currently, most of the available human resources are fully employed at the lower levels of tedious data interpretation. Third, there is an increasing need for more rapid and accurate decision-making capabilities. Typically, commanders and their staff find themselves in continuous replanning mode as the most carefully laid plans require significant adjustments due to unforeseen events that will inevitably occur during implementation.

The response to these requirements has been the desire to create a knowledge management environment. So, what is knowledge management? Knowledge is an intellectual facility that allows a person to perform tasks that require an understanding of what has to be accomplished, the formulation of a plan of action, and the skills that are required to undertake the task. It normally involves the acquisition over time of factual information, associations that bind the factual information into more general patterns, principles, rules, and problem solving skills. A person acquires knowledge through experience, formal education, and a life-long process of self-education. Accordingly, knowledge is a commodity that is held within the brain of each individual person. Both the communication of this personal knowledge from one individual to another and the collection of the knowledge as a corporate asset has become a serious concern of organizations, and is commonly referred to as knowledge management.

The implementation of the Infostructure-GIG vision as a knowledge management environment will require: (1) the formalization of communities of interest; (2) the standardization of nomenclature and reference tables; (3) the definition of logical data models; (4) the publication of data encoding protocols and formats (i.e., metadata registries); (5) the adoption of data transmission standards; (6) the development of functionally created ontologies that allow data to be placed in context; (7) the publication of business rules and the encoding of these rules in software agents with automated reasoning capabilities; and, (8) the formulation of policies, conventions, and processes, designed to facilitate planning, problem solving, and decision-making tasks.

What is the appropriate architecture for the realization of the Infostructure-GIG vision? To answer this question we need to first consider the products of the current data-processing environment, because these products will continue, at least for the near term, to serve as the principal data sources of the new knowledge management environment. Since the early 1970s the ability of computers to store large amounts of data has been increasingly exploited by industry and government. The potential bottleneck presented by these electronic data stores did not become apparent until more recent times with the increasing desire and expectation that their contents should be utilized for planning and decision making purposes. The need to integrate and analyze data from multiple sources led to the concept of a Data Warehouse that is updated periodically, usually with summarized data collected from operational data sources. Structured into compartments or Data Marts, each focused on a particular functional area, the Data Warehouse serves as a basis for analyzing historical trends with On Line Analytical Processing (OLAP) tools and projecting future conditions with Data Mining tools. However, the usefulness of these tools is greatly constrained by lack of context. Even though the data in Data Warehouses are typically stored in relational databases, they commonly contain few relationships that are focused on the exploitation of the data in a particular functional domain. Therefore, the ability of OLAP and Data Mining tools to answer What?, Why? and What-if? questions is severely constrained by the very limited context provided by the data.

Where the operational data sources are of good quality and can be used directly for decision-making purposes, gateways have been implemented in recent times to provide convenient access to disparate data sources. These gateways are referred to as Data Portals and do not in themselves store data. Apart from accessing the data sources the principal functions of such Portals include the presentation of data to the user. Some Data Portals also include data analysis tools aimed at enriching the presentation capabilities.

Data Portals and Data Warehouses represent a structured data level that integrates the multiple, fragmented databases, files, documents, and e-mail messages that constitute the often only moderately organized operational data flow. By providing access to both the operational data (Data Portals) and the archived summary data (Data Warehouses) this structured data level represents the integrating Data Layer that constitutes the bottom layer of a knowledge management system, serving as a necessary foundation for an upper Information Layer (Figure 1). The upper layer utilizes one or more internal information models or ontologies (Figure 2) to provide context for the automatic reasoning capabilities of software agents.

What is an ontology? The term ontology is loosely used to describe an information model, rich in relationships that focuses on the utilization (rather than the taxonomic structure) of data. An ontology, therefore, is a virtual representation of some real world environment that provides

context (e.g., the management of a transport corridor, the loading of a cargo ship, the coordination of a military theater, the design of a building, and so on). The elements of an ontology include objects and their characteristics, different kinds of relationships among objects, and the concept of inheritance. Ontologies are also commonly referred to as object models. However, strictly speaking the term ontology has a much broader definition. It actually refers to the entire knowledge in a particular field. In this sense an ontology would include both an object model and the software agents that are capable of reasoning about information within the context provided by the object model.

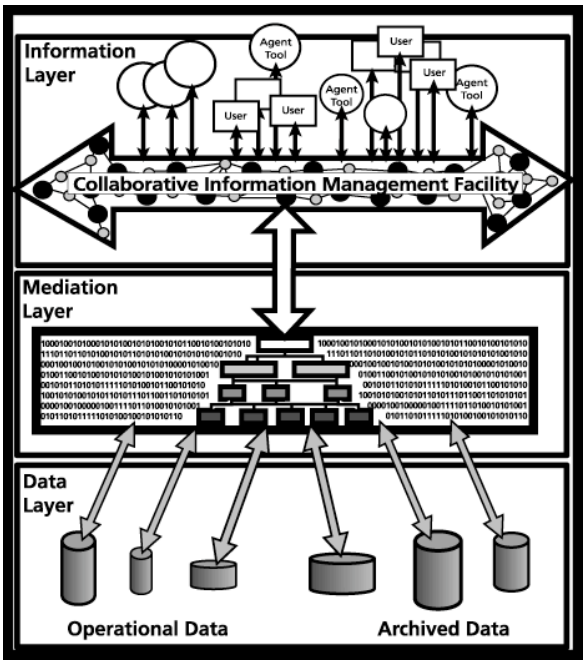


Figure 1: Schematic architecture of a knowledge management system

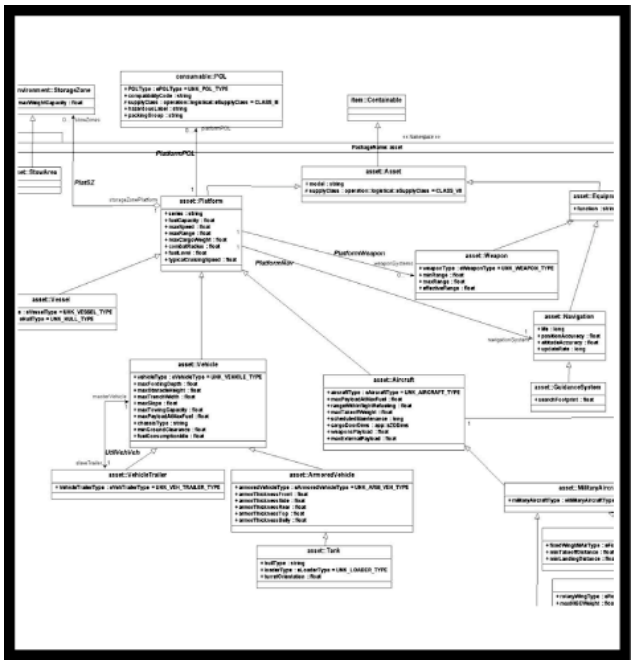


Figure 2: Portion of a typical information model (ontology) in the logistic domain

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

Referring back to Figure 1, the interface between the lower data-processing layer and the higher information management layer consists of a translation facility that is capable of mapping the data schema of the lower layer to the information representation (i.e., ontology) of the upper layer. In this manner, the ontology of the Information Layer can be populated with near real-time operational data and archived summary data from Data Warehouses. This mapping process should be bidirectional so that the results of agent actions can be readily transmitted to any data-centric applications that reside in the Data Layer. The interface level that supports this critical process is now commonly referred to as the Mediation Layer. Within this architecture the Mediation Layer will include not only metadata registries, but also a powerful suite of generic

information services can be used by communities of interest to maintain their metadata and comply with enterprise-wide requirements in information management areas such as security, privacy, and application software installation and configuration policies. These generic computing services should also include software tools that will allow the automated verification of the compliance of ontologies with their underlying logical data models and standardized nomenclatures. Conversely, GES computing services could also assist application developers by generating a base set of ontology elements from the appropriate domain of a logical data model. The executive agent for GES in the DoD community is the Defense Information Systems Agency (DISA).

Finally, what is metadata? Often succinctly defined as data about data, metadata includes the descriptions that define the organization of data and information so that the interpretation of such data and information can be undertaken automatically by computer software. Metadata typically includes specifications for nomenclature (i.e., vocabulary), the structure of data in logical data models, taxonomies, interfaces, mapping tables, object models (i.e., ontologies), and business rules. The DoD Net-Centric Data Strategy (Stenbit 2003, 6-8) describes three principal mechanisms for storing and processing metadata, as follows. Metadata Registries are used to describe the structure, format, and definition of data. They may be implemented as a software application that uses a database to facilitate the storing and searching for data, definitions of data, relationships among data, and document formats. In this respect a Metadata Registry is like a library document that defines the kind of information that is required to be printed on the cards of a library card catalog (i.e., it does not describe any particular library holding, but only the kind of information that needs to be provided for all holdings). A Metadata Catalog, on the other hand, provides the information stipulated by the Metadata Registry for each individual set of data. This information is also typically stored in a database. Shared Space refers to the storage of the actual data that are described in the Metadata Catalog. In the three-layer architecture of a typical knowledge management system shown in Figure 1, the Metadata Registries and Catalogs reside in the Mediation Layer while the actual data (i.e., Shared Spaces) is found in the Data Layer.

Jens Pohl, June 2004

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

Stenbit, J.; 'Department of Defense Net-Centric Data Strategy'; US Department of Defense, Chief Information Officer, Wash., DC, May 9, 2003.

InterSymp-2004
International Conference on Systems Research, Informatics and Cybernetics

Focus Symposium
on

Intelligent Software Systems for the New Infostructure

Friday, July 30, 2004

TABLE OF CONTENTS

Section 1:

Information-Centric Technology Underpinnings

The Evolution of Intelligent Computer Software and the Semantic Web

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 trace the evolution of intelligent software from data-centric applications that essentially encapsulate their data environment to ontology-based applications with automated reasoning capabilities. The author draws a distinction between human intelligence and component capabilities within a more general definition of intelligence, which may be embedded in computer software. The primary vehicle in the quest for intelligent software has been the gradual recognition of the central role played by data and information, rather than the logic and functionality of the application. The three milestones in this evolution have been: the separation of data management from the internal domain of the application; the development of standard data exchange protocols such as XML that allow machine interpretable structure and meaning to be added to data exchange packages; and, the ability to build information models that are rich in relationships and are thereby capable of supporting the automated reasoning capabilities of software agents.

The author suggests that the vision of a Semantic Web environment in which ontology-based Web services with intelligent capabilities are able to discover each other and individually or in self-configured groups perform useful tasks, is not only feasible but imminently realizable. The capabilities of an experimental proof-of-concept system featuring semantic Web services that was demonstrated at an Office of Naval Research Workshop in 2002 is described in summary form.

Keywords

agents, artificial intelligence, data-centric, information-centric, Internet, object model, ontology, Semantic Web, semantic Web services, XML

The Concept of 'Intelligence'

Before we can proceed with the theme of this paper it is necessary to briefly discuss the concept of ***intelligence*** and the sense in which this concept is applied by the author. There are those that have advanced strong arguments that intelligence is the province of living creatures and that machines, such as electronic computers, do not and will never display any truly intelligent capabilities (Dreyfuss 1979 and 1997, Dreyfuss and Dreyfuss 1986, Lucas 1961, Searle 1980 and 1992). In most cases these arguments are based on the premise that intelligent behavior is closely associated with the human body and mind, and that the powerful notions of common sense and intuition are essential ingredients of intelligence. It is not the purpose of this paper to attempt to counter these arguments or even take sides in this debate.

Instead, the author wishes to advance another view of intelligence, namely that human intelligence and intelligence are not synonymous. We human beings are a decidedly self-centered species. We tend to view our capabilities and our interactions with our surroundings from a very personal point of view. It is therefore not surprising that we should consider intelligence, which is essentially our most powerful asset, to be restricted to living creatures among whom we believe ourselves to reign supreme.

Webster's Dictionary (Random 1999) defines intelligence as the "… capacity for learning, reasoning, and understanding;". This definition suggests that there are component capabilities that contribute to the concept of intelligence. Further, these component capabilities are not necessarily equally powerful. In other words, it may be argued that there are levels of intelligence and that at the lowest level such capabilities must include at least the ability to remember. Higher levels of intelligence include reasoning, learning, discovering, and creating. Certainly at least some of these intelligent capabilities can be embedded in computer software. For example, computers excel at storing and recalling data in virtually unlimited quantities and over very long periods of time. Computers can reason about data quite effectively, if adequate context is made available with the data. Also, computers have been shown to have learning-like capabilities, and computers can discover information through associations and pattern matching.

There is no intention by the author to suggest that computer intelligence is equal or even similar to human intelligence, but rather that computer intelligence and human intelligence may be applied in parallel to complement each other. Furthermore, a strong case can be made in support of the view that there is an urgent need for intelligent computer capabilities due to the mounting expectations of accuracy, quality and timeliness in a globally connected environment of rapidly increasing complexity.

The Need for Software Intelligence

There are essentially two compelling reasons why computer software must increasingly incorporate more and more 'intelligent' capabilities. The first reason 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.

The second reason is somewhat different in nature. It relates to the complexity of networked computer and communication systems, and the increased reliance of organizations on the reliability of such information technology environments as the key enabler of their effectiveness, profitability and continued existence.

The Data-Processing Bottleneck

This requires further explanation, as a fundamental issue and one of the primary forces driving the evolution of software intelligence. The design of any information system architecture must be based on the obvious truth that the only meaningful reason for

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

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

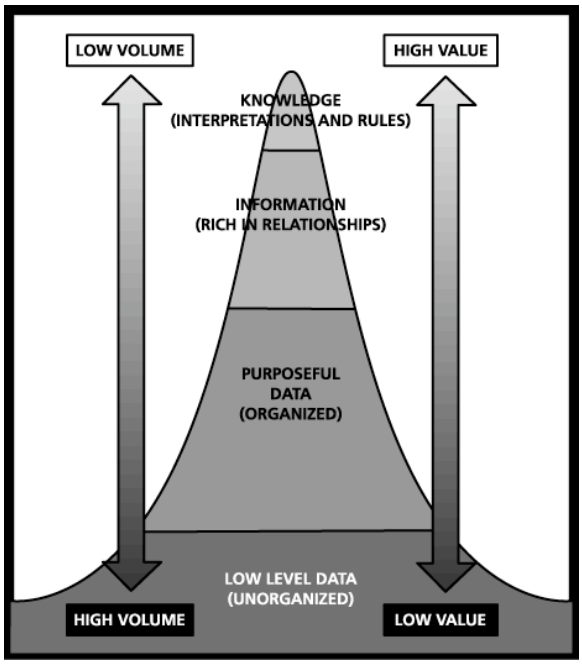


Figure 1: Transition from data to knowledge

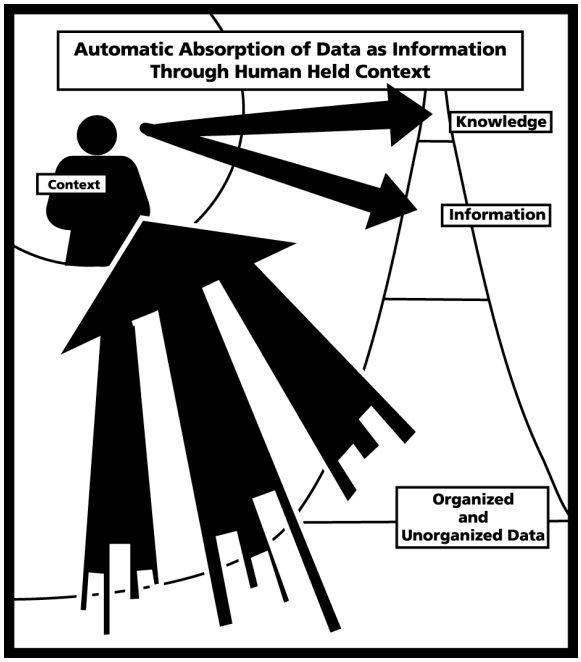


Figure 2: Human interpretation of data

The reasons for this bottleneck are twofold. First, large organizations are forced to focus their attention and efforts on the almost overwhelming tasks involved in converting unordered data into purposefully ordered data (Figure 1). This involves, in particular, the establishment of gateways to a large number of heterogeneous data sources, the validation and integration of these sources, the standardization of nomenclatures, and the collection of data elements into logical data models. Second, with the almost exclusive emphasis on the slicing and dicing of data, rather than the capture and preservation of relationships, the interpretation of the massive and continuously increasing volume of

data is left to the users of the data (Figure 2). The experience and knowledge stored in the human cognitive system serves as the necessary context for the interpretation and utilization of the ordered data in monitoring, planning and decision-making processes. However, the burden imposed on the human user of having to interpret large amounts of data at the lowest levels of context has resulted in a wasteful and often ineffective application of valuable and scarce human resources. In particular, it often leads to late or non-recognition of patterns, overlooked consequences, missed opportunities, incomplete and inaccurate assessments, inability to respond in a timely manner, marginal decisions, and unnecessary human burn-out. These are symptoms of an incomplete information management environment. An environment that relies entirely on the capture of data and the ability of its human users to add the relationships to convert the data into information and thereby provide the context that is required for all effective planning and decision-making endeavors.

A more complete information management environment considers data to be the bottom layer of a three-layer architecture, namely:

> A ***Data Layer*** that integrates heterogeneous data sources into accessible and purposefully ordered data. It typically includes a wide variety of repositories ranging from simple textual files to databases, Data Portals, Data Warehouses, and Data Marts.
>
> A ***Mediation Layer*** that defines the structure of the data sources (i.e., logical data models), data transfer formats, and data transformation rules. The two principal purposes of the Mediation Layer are to facilitate the automated discovery of data and to support the mapping of data to information. In other words, the Mediation Layer serves as a registry for all definitions, schemas, protocols, conventions, and rules that are required to recognize data within the appropriate context. The Mediation Layer also serves as a translation facility for bridging between data with structural relationships (e.g., based on a logical data model) and information that is rich in contextual relationships.
>
> An ***Information Layer*** that consists of many functionally oriented planning and decision-assistance software applications. Typically, these applications are based on internal information models (i.e., object models or ontologies) that are virtual representations of particular portions of the real world context. By providing context, the internal information model of each application is able to support the automated reasoning capabilities of rule-based software agents.

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

importantly, the software agents continuously and tirelessly monitor the real world execution environment for changes and events that may impact current or projected plans.

The Increasing Complexity of Information Systems

The economic impact on an organization that is required to manually coordinate and maintain hundreds of interfaces between data-processing systems and applications that have no 'understanding' of the data that they are required to exchange, is enormous. Ensuing costs are not only related to the requirement for human resources and technical maintenance (normally contracted services), but also to the indirect consequences of an information systems environment that has hundreds of potential failure points.

Recent studies conducted by IBM Corporation and others have highlighted the need for autonomic computing as the organizational expectations and dependence on information services leads to more and more complex networked computer solutions (Ganek and Corbi 2003). In the commercial sector "…it is now estimated that at least one-third of an organization's IT (Information Technology) budget is spent on preventing or recovering from crashes" (Patterson et al. 2002). Simply stated (Figure 3), autonomic computing utilizes the 'understanding' that can be represented within an information-centric software environment to allow systems to automatically: (1) reconfigure themselves under dynamically changing conditions; (2) discover, diagnose, and react to disruptions; (3) maximize resource utilization to meet end-user needs and system loads; and, (4) anticipate, detect, identify, and protect themselves from external and internal attacks.

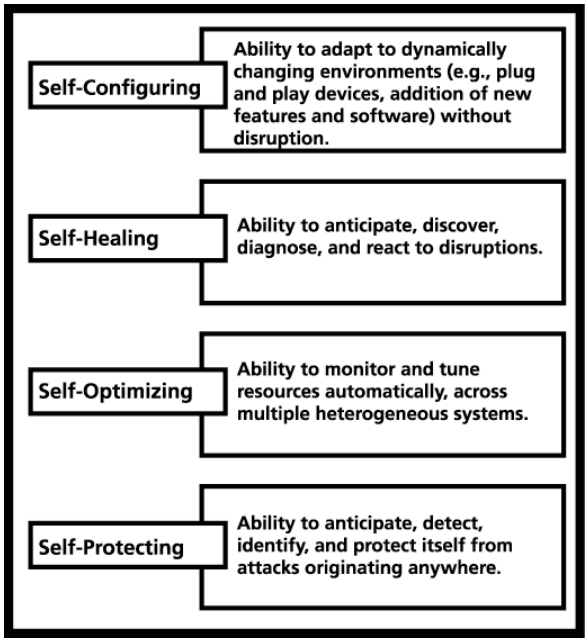


Figure 3: Desirable autonomic capabilities

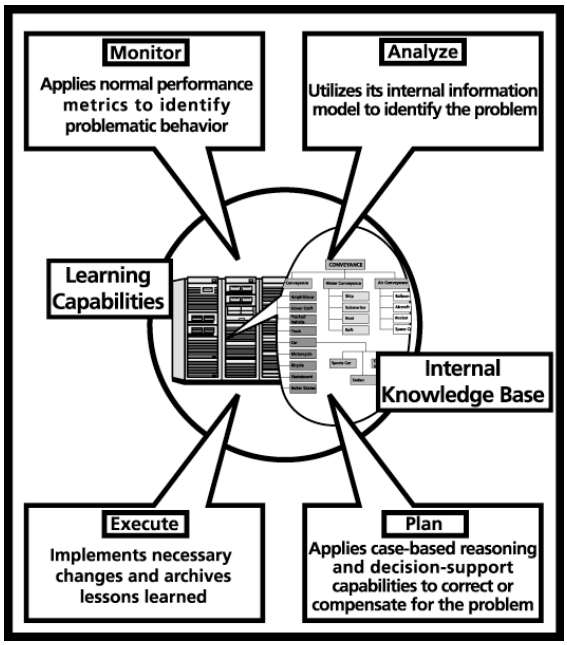


Figure 4: Autonomic self-healing requirements

These same studies have found that more than 40% of computer system disruptions and failures are due to human error. However, the root cause of these human errors was not found to be lack of training, but system complexity. When we consider that computer 'downtime' due to security breaches and recovery actions can cost as much as (US)$2

million per hour for banks and brokerage firms, the need for computer-based systems that are capable of controlling themselves (i.e., have autonomic capabilities) assumes critical importance.

A core requirement of autonomic computing is the ability of a computer-based information system to recover from conditions that already have caused or will likely cause some part(s) of the system to fail. As shown in Figure 4, this kind of self-healing capability requires a system to continuously monitor itself so that it can identify, analyze and take mitigating actions, preferably before the disruption takes place. In addition, the system should be able to learn from its own experience by maintaining a knowledge base of past conditions that have caused malfunctions and the corrective measures that were taken.

In summary, the continued expansion of networks (e.g., the Internet and its successors) will provide seamless connectivity among countless nodes on a global scale. While the collection of data has already increased enormously over the past decade, the availability of such a global network is likely to increase the volume of data by several orders of magnitude. Such a volume of raw data is likely to choke the global network regardless of any advances in communication and computer hardware technology. To overcome this very real problem there is a need to collect data in context so that only the data that are relevant and useful are collected and transmitted within the networked environment. Most (if not all) of the necessary filtering must be achieved automatically for at least three reasons. First, organizations cannot afford to utilize human resources for repetitive tasks that are tedious and require few human intellectual skills. Second, even if an organization could afford to waste its human resources in this manner it would soon exhaust its resources under an ever-increasing data load. Third, it does not make sense for an organization to 'burn-out' its skilled human resources on low-level tasks and then not have them available for the higher-level exploitation of the information and knowledge generated by the lower level tasks.

Finally, the increased reliance on computer-based information systems mandates a level of reliability and security that cannot be achieved through manual means alone. The alternative, an autonomic computing capability, requires the software that controls the operation of the system to have some understanding of system components and their interaction. In other words, autonomic computing software demands a similar internal information-centric representation of context that is required in support of the knowledge management activities in an organization. In both cases the availability of data in context is a prerequisite for the reasoning capabilities of software agents (i.e., the automatic interpretation of information by the computer).

A Framework for Assessing Software Capabilities

Just like the initial conception and implementation of computing devices was driven by the human desire to overcome the limitations of manual calculation methods, the advancements in computing technology during the past 50 years have been driven by the desire to extend the usefulness of computer-based systems into virtually every human activity. It is not surprising that after several orders of magnitude increases in hardware performance (i.e., computational speed and data storage capacity (Pohl 1998)) had been achieved, attention would gradually shift from hardware to software.

Increasingly software is being recognized as the vehicle for computers to take over tasks that cannot be completely predefined at the time the software is developed. The impetus for this desire to elevate computers beyond data-processing, visualization and predefined problem-solving capabilities, is the need for organizations and individuals to be able to respond more quickly to changes in their environment. Computer software that has no 'understanding' of the data that it is processing must be designed to execute predefined actions in a predetermined manner. Such software performs very well in all cases where it is applied under its specified design conditions and performs increasingly poorly, if at all, depending on how much the real world conditions vary from those design specifications. Instead, what is needed is software that incorporates tools, which can autonomously adapt to changes in the application environment.

Adaptable software presupposes the ability to perform some degree of automated reasoning. However, the critical prerequisite for reasoning is the situational context within which the reasoning activity is framed. It is therefore not surprising that the evolution of computer software in recent years has been largely preoccupied with the relationship between the computational capabilities and the representation of the data that feed these capabilities. One could argue that the historical path from unconnected atomic data elements, to data structures, relational databases, data objects, object-oriented databases, object models, and ontologies, has been driven by the desire to provide information context in support of automated reasoning capabilities.

However, to be able to present a true historical perspective of the evolution of software it is necessary to take into account a more comprehensive set of criteria. In fact, there are several factors that have in the past and are continuing to contribute to the evolution of intelligent software. This section will attempt to establish a set of categorization criteria to serve as a framework for tracing the capabilities of software. Since these capabilities are closely related to the design and implementation of the computer-based environment within which the software is required to operate, the proposed framework will utilize ***system architecture*** as a yardstick and milestone component. The following eight system architectures have been selected to serve as milestones for the assessment of software capabilities:

- ***Single data-centric applications*** that operate in a stand-alone mode and receive data from user interaction and other closely coupled sources (e.g., data files and dedicated databases).
- ***Confederation of linked data-centric applications*** with application-to-application data bridges. Also described as 'stove-pipe' systems because the system components are essentially hardwired to only work together within their confederation.
- ***Shared database systems*** consisting of multiple data-centric applications that are able to share data between themselves and a common repository, through application-to-database bridges. The repository may be either a single database or a distributed database facility.
- ***Distributed expert systems*** with dedicated knowledge bases (i.e., rules) and a single shared fact list (i.e., data).
- ***Distributed static information-based applications*** with collaborative agents, capable of exchanging data with external data-centric applications.
- ***Distributed static information-sharing applications*** with collaborative agents, capable of interoperating at the 'information' level with other ontology-based applications and capable of exchanging data with external data-centric applications.

- ***Distributed extensible information-sharing applications*** with collaborative agents, capable of interoperating at the 'information' level with other ontology-based applications and capable of extending their internal information representation (i.e., ontology) during execution.
- ***Semantic Web services*** capable of discovering other Web services and dynamically configuring themselves into distributed systems on an as-needed basis.

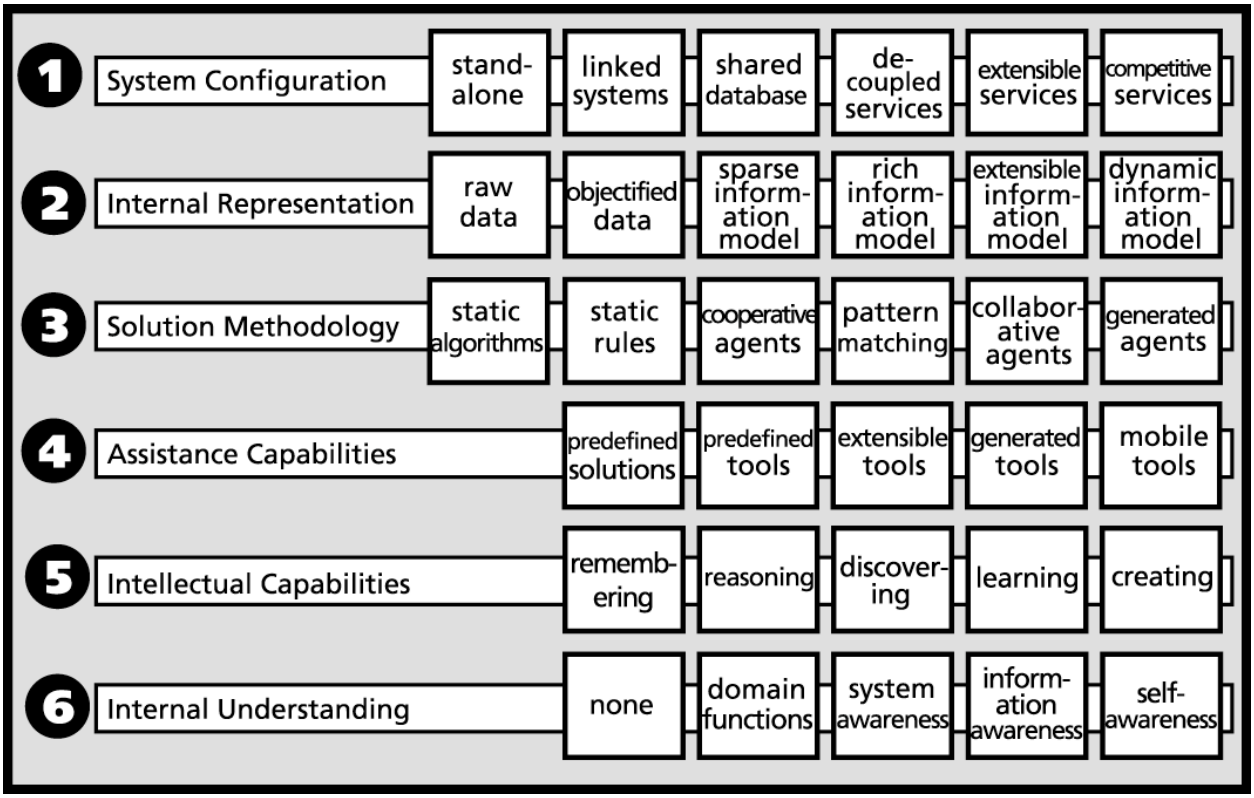


Figure 5: Software characterization categories and their capability criteria

The software capabilities that have been in the past or are still today prevalently applied in each of these system architectures are characterized within six capability groups as shown in Figure 5. While the first of these groups (i.e., Group (1) *System Configuration*) is intended to describe principal architectural features, the other five groups are focused on the degree to which the software is capable of representing and processing data with or without context in partnership with the human user. Fundamental in this respect is Group (2) *Internal Representation.* The manner in which an application represents the data that it is intended to manipulate essentially determines the level of software intelligence that the application is capable of supporting. Group (2) differentiates among applications that represent data without context (i.e., 'raw data' and 'objectified data'), applications that provide context in the form of a static information model (i.e., sparse information model' and 'rich information model') and applications with information models that are extensible during execution (i.e., 'extensible information model' and 'dynamic information model'). The remaining four groups address the general solution methodology available to the application, its decision-support capabilities, and the level of internal 'understanding' of its capabilities, activities and intrinsic nature. The divisions within each of the

groups will be defined in more detail during the discussion of each of the eight system architectures.

The first system architecture for discussion (Figure 6) is representative of the typical early computer applications, namely a stand-alone application that receives all of its data from the user and/or data sources that are considered to be part of the application. Whether or not the data are treated as discrete elements or objects, the *Internal Representation* includes only a very limited set of relationships and therefore lacks context. Under these circumstances the *Assistance Capabilities* are limited to predefined solutions utilizing static algorithms, no internal understanding can be provided by the representation of data without relationships, and the *Intellectual Capabilities* of the software are restricted to 'remembering' since the data are stored in the computer. The second system architecture (Figure 7) adds data bridges between several data-centric applications. Each bridge is simply an application-to-application mapping of the data format of one application to the other. Therefore, the only capability that this architecture adds to the previously discussed architecture is that the *System Configuration* supports a confederation of tightly linked applications.

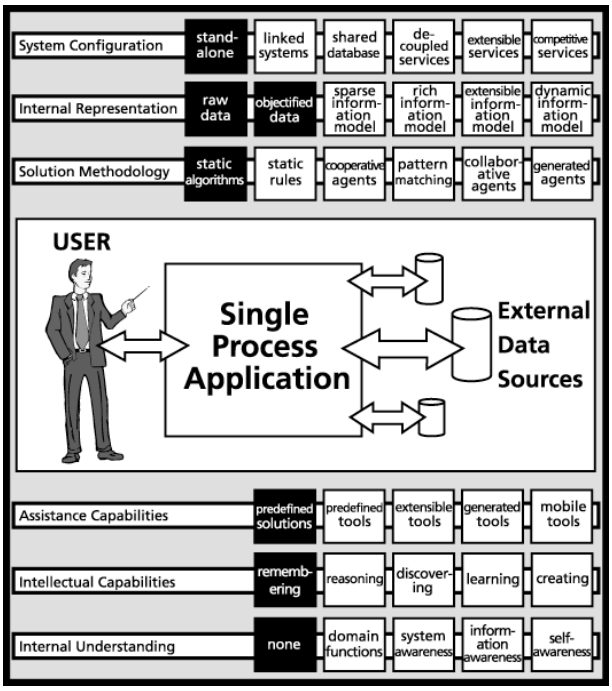


Figure 6: Single data-centric applications

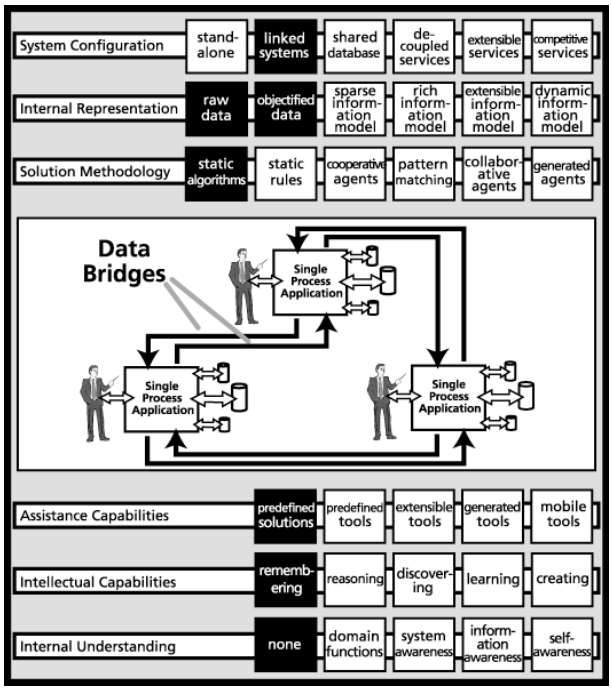


Figure 7: Confederation of linked data-centric applications

The shared database architecture (Figure 8) constitutes a major improvement over the first two system architectures by separating the data from the application and placing the former into a common repository that is external to all of the applications. The recognition that data and not the application should be the dominant component of a data-processing environment sets the stage for interoperability and intelligent software. However, it does not directly contribute any additional capabilities to the software criteria. The reason is the absence of data context, and this applies equally to the three system architectures discussed so far.

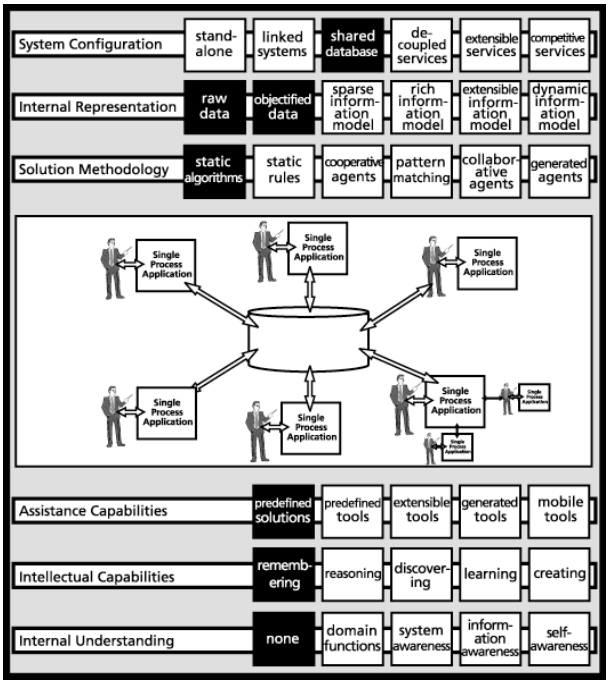


Figure 8: Shared database systems

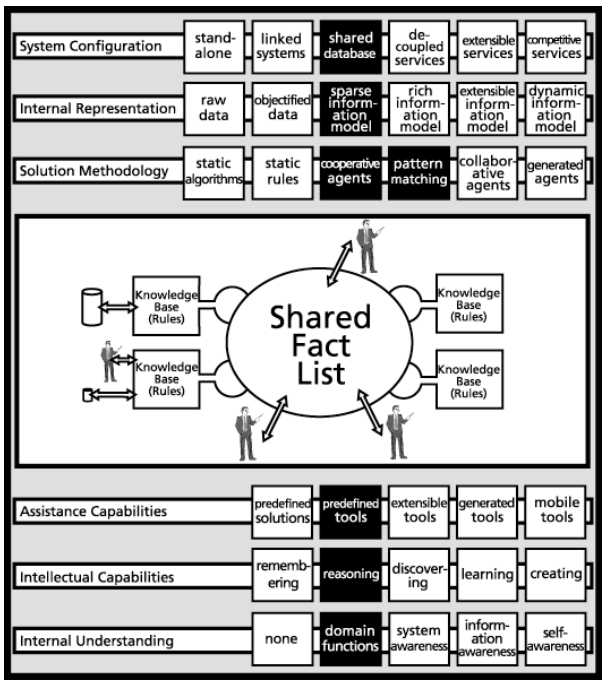


Figure 9: Distributed expert systems

The distributed expert system architecture shown in Figure 9 on the other hand, by virtue of its internal knowledge base of rules, driven by a shared repository of facts, adds several new capabilities to the software. Each knowledge base provides relationships and therefore represents a local component of what might be characterized as a sparse information model. This model provides adequate support for some form of automated reasoning within the typically narrow domain of each expert system. Although the expert systems (or agents) now operate as tools rather than predetermined solutions, their rules are nevertheless predefined and typically not extensible during execution.

For at least two reasons the concept of expert systems represents a milestone in the transition from data-processing to information-centric software. First, it showed that automated rule-based reasoning is in fact feasible and thereby allowed the field of artificial intelligence to regain some confidence after its earlier failures. Second, the largely opportunistic pattern-matching nature of an expert system laid the foundations for the notion of demon-like modules with particular data interests that could be triggered into action by data changes. Over the next decade these modules developed into *flexible* software agents that are *situated* in some environment and capable of *autonomous* actions (Wooldridge and Jennings 1995, Pohl et al. 2001 (32-33)). It was highly desirable for these agents to be capable of acting without the direct intervention of human users (or other agents), thereby providing the system with some degree of control over its own actions and internal state. The ability to achieve this level of autonomous behavior was greatly facilitated by situating the agent in a sufficiently well represented environment, which it can monitor and act upon. Triggered by its environment the agent is then able to respond to changes in the environment, exercise intiative through goal-directed reasoning capabilities, and utilize the services of other agents (including the human user) to supplement its own problem-solving capabilities in a collaborative fashion.

The desire for software agents to perform increasingly more valuable and human-like reasoning tasks focused a great deal of attention on the virtual representation of the real world environment in which the agent is situated. It became clear that the reasoning capabilities of a rule-based software agent depend largely on the richness of the virtual representation of this physical and conceptual environment. Taking advantage of the capabilities of object-oriented languages, which allow objects to be represented as classes with attributes and relationships, a new generation of application software with internal object-based information models was born (Figures 10, 11 and 12). These are often referred to as ontology-based applications and are typically distributed in nature.

It should be noted that the term ontology is commonly used rather loosely as a synonym for object model. Strictly speaking, however, the term ontology has a much broader definition. It actually refers to the entire knowledge in a particular field. In this sense, an ontology includes both an object model and the software agents that are capable of reasoning about information within the context provided by the object model (i.e., since the agents utilize business rules, which constitute some of the knowledge within a particular domain). In this paper the common use of the term ontology as an object model (i.e., context) is implied.

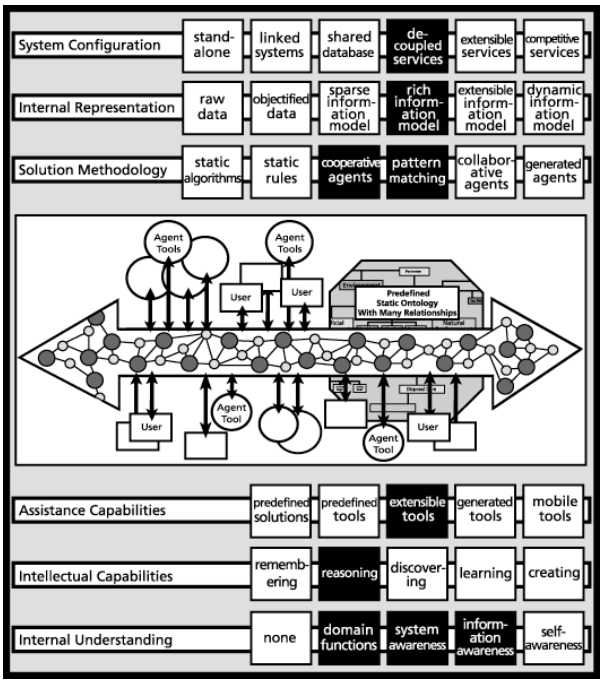


Figure 10: Information-based applications

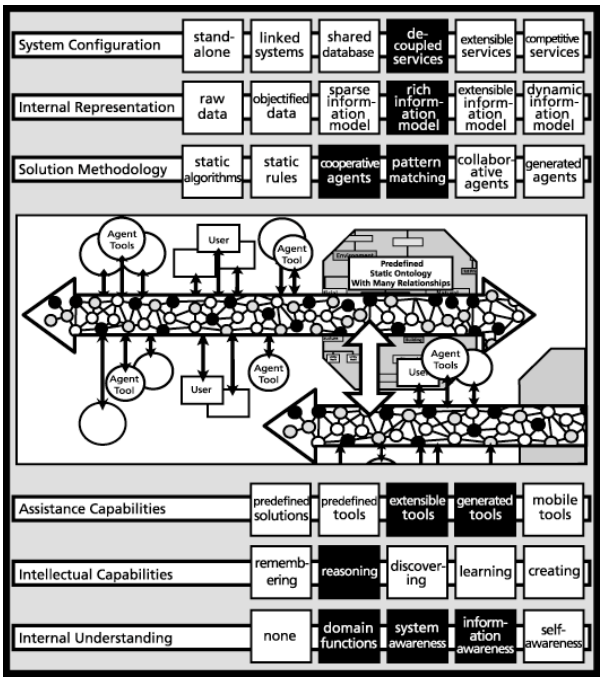


Figure 11: Information-sharing applications

The information-based architecture shown in Figure 10 typically consists of components (e.g., agents and user-interfaces) that communicate with each other through an information-serving collaboration facility. Each component includes a relevant portion of the ontology and a subscription profile of the kind of information that it is interested in receiving from this facility. Since the components have at least a limited understanding of the real world situation only the changes in the situation need to be communicated to them. While the existence of a subscription service obviates the need for computationally expensive queries in most cases, the ability to restrict the communication to changes in information also greatly reduces the amount of data that has to be exchanged. This applies equally to the information-sharing architecture and the

extensible information architecture shown in Figures 11 and 12, respectively. Also, in all three of these software architectures system capabilities support (and promote) decoupled applications that interact via these services, which are accessed internally through clearly defined interfaces. Apart from simplifying the design and development of such applications, this allows services to be seamlessly replaced as long as the replacement service adheres to the same interface definition.

The principal differences among these three architectures are related to the adaptability and accessibility of the ontology within each of the information-centric systems. First, in both the information-based (Figure 10) and the information-sharing (Figure 11) architectures the ontologies are predefined at the time the applications are compiled and cannot be changed during execution. While it is certainly possible to build into an ontology some degree of flexibility that allows for the definition of variations of existing object types during execution, the context-based definition of new objects requires the application to be recompiled. In other words, the ontology is essentially static after the application has been compiled. In the extensible information-sharing architecture shown in Figure 12, an application is able to gain and share knowledge in its interactions with other applications that have similar capabilities, or with human users. The ability of an application to extend its understanding (i.e., to increase the context within which its agents are able to reason about changes in the real world situation) is still largely a subject of research. It involves the construction of context from data with sparse relationships, which intuitively would appear to be a poor approach. However, utilizing lexical (Fellbaum 1998) and algorithmic approaches developed in the natural language research domain (Pedersen and Bruce 1998), some surprisingly promising progress has been made in this area in the commercial arena (Cass 2004).

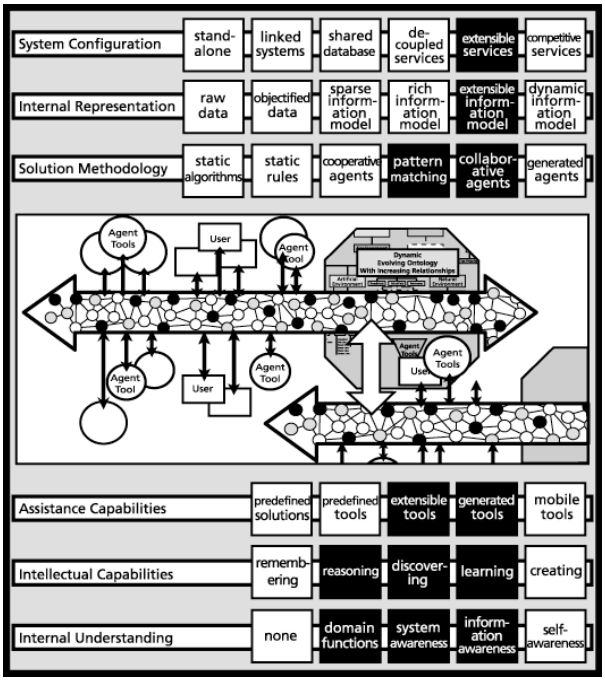


Figure 12: Extensible information-sharing applications

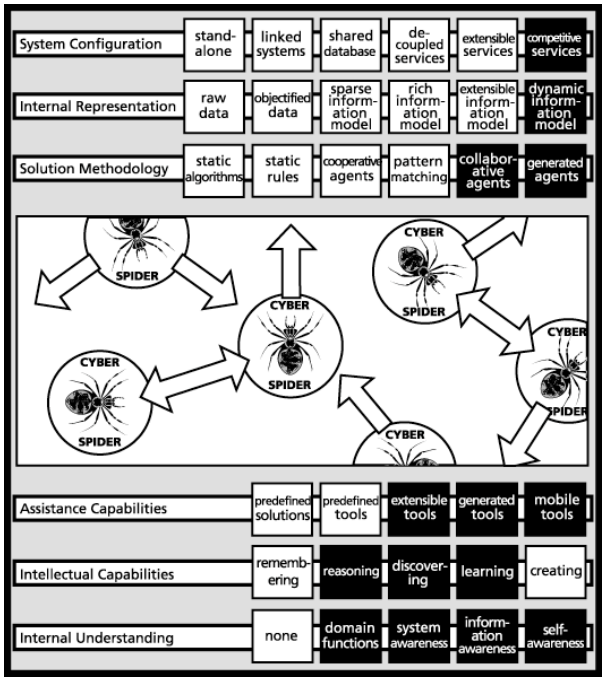


Figure 13: Semantic Web services

Second, in terms of accessibility, the subscription capabilities embedded in the components of an information-based system can be equally applied across multiple systems by having the information-serving collaboration facility of one system subscribe to the information-serving collaboration facility of another system. This is potentially a very powerful approach that allows information-centric systems to scale as clusters of networks within a networked environment.

The software architectures described so far (i.e., Figures 6 to 12) progressively evolved from stand-alone systems that encapsulate their own data, to systems that are able to share data based on predefined formats for data representation, to systems that incorporate rich but static information models and are able to support automated reasoning capabilities, to systems that are able to extend their internal information models in collaboration with similar ontology-based external systems. Within this evolutionary path the transition from data-based to information-based internal representation schemas is the enabling step that has endowed software with increasingly intelligent capabilities. However, the fundamental mechanism for achieving these capabilities is the ability to automatically reason about changes in the current state of the situation described by the information model. Once expert systems (Figure 9) had demonstrated that reasoning capabilities could be provided by conditional rules (i.e., a knowledge base of productions) and triggered by changes in a simple fact-list, it became clear that much could be gained by expanding the representational capabilities of the fact-list and incorporating in it many of the relationships that were formerly encoded in the rules of the knowledge base. This contributed to the formal separation within an application of the representation (i.e., object model or ontology) and the logic that is applied to this representation by agents. While initially most of the complexity of these ontology-based applications continued to reside in the agents, the availability of more powerful modeling concepts and tools is gradually allowing more and more of the complexity to be moved from the agents into the ontology. This suggests a trend that appears to mirror the earlier separation of an application from the data it is designed to manipulate (Figure 8), namely the separation of the information representation from the applications that incorporate reasoning capabilities. The combination of this trend with an information-centric Internet-like environment will cast applications into the role of capability-based services.

This is the emerging concept portrayed by the semantic Web services architecture shown in Figure 13. However, before describing this software architecture it is necessary to briefly discuss the architecture and capabilities of the existing data-centric Web services. They typically comprise a Web-Server that utilizes the Hyper-Text Transfer Protocol (HTTP) for communication, the Universal Description Discovery and Integration (UDDI) protocol as part of the standard definition of Web services registries, and a Registry that already contains an entry for the accessing application as well as any number of other Web services. UDDI is an international standard that defines a set of methods for accessing a Registry that provides certain information to an accessing application. For perhaps historical reasons UDDI 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).

The Simple Object Access Protocol (SOAP) defines a protocol for the direct exchange of data objects between software systems in a networked environment. It provides a means of representing objects at execution time, regardless of the underlying computer language. SOAP defines methods for representing the attributes and associations of an object in the Extensible

Markup Language (XML). It is actually a meta-protocol based on XML that can be used to define new protocols within a clearly defined, but flexible framework.

Web-Services are designed to be accessed by software. In the currently prevalent data-centric software environment they are generally clients to the middleware of data sources. The middleware collects the required data and sends them back to the Web service, which reformats the data using the SOAP protocol and passes them onto the requester. Depending on its original specifications, the requesting application will have the data downloaded on disk or receive them directly on-line. If the Web service is a data-centric application then a data-to-data translation must be performed in much the same way as is necessary when passing data between two data-centric applications.

Returning to the software architecture shown in Figure 13, the emphasis is on the word ***semantic***. In this architecture the semantics are embedded in an ontology, which provides the necessary context for automated reasoning. A semantic Web service, therefore, is an ontology-based application (may be mobile) with certain capabilities. Given a particular intent it seeks the services that it determines to be necessary for satisfying this intent. Having found one or more such Web services it self-configures itself with these discovered services into a temporary system. Depending on needs and circumstances this transitory system may reconfigure itself by discarding existing members when their capabilities are no longer needed, adding new members when other requirements arise, or dissolving itself altogether once it determines that its intent has been adequately executed.

To meet these capability objectives a semantic Web service reaches the highest-level criteria in all but one of the six software characterization categories shown in Figures 5 and 13. First, it operates in a competitive environment where it can select a service from several offering candidates, and presumably negotiate the terms of acceptance. Second, it incorporates a rich and extensible information model that will change dynamically as the semantic Web service discovers, collaborates with, and shares ontology fragments with its transitory partners. This provides the ability to create and maintain a desirable degree of common understanding within the self-configured system. Third, by virtue of this common understanding the agents of each member of the system are able to collaborate beyond the boundaries of the particular semantic Web service that they are housed in. Furthermore, any new agents that may be generated in response to a recently emerged need will likewise be able to collaborate globally within the system.

Forth, the agents, which constitute the primary assistance capabilities of the system, become highly adaptable tools. They are extensible, they may be generated dynamically during execution to satisfy emerging new needs, and they can be implemented to operate in a mobile mode. Fifth, the collective intellectual capabilities of the system include the ability to discover capabilities that may be made available by external services and the ability to increase its understanding of context by extending the ontologies of one or more of its members through their interaction and the addition of new members to the system. It can be argued that this dynamic acquisition of new knowledge is a form of learning, however, it does not necessarily imply an ability to create new knowledge. Whether or not the semantic Web architecture will be able to create new knowledge is very much a matter of conjecture at this time.

Finally, in the *Internal Understanding* category the semantic Web architecture is rated to have the potential for reaching the highest criterion, ‘self-awareness’. As further explanation it should be noted that this characterization category has been based entirely on the representational

capabilities of ontologies, since the author is not aware of any alternative method for creating internal understanding in software. Ontologies are capable of not only representing physical objects such as buildings, conveyances (e.g., cars, boats, aircraft), supplies, weapons, and organizations, but also conceptual objects such as the notions of mobility, threat, privacy, security, consumability, and so on. This has been the predominant focus of ontologies to date. However, in addition, ontologies are able to represent the behavioral characteristics and relationships of the components of the software system itself. This is the domain of autonomic computing discussed previously, whereby a system is charged with continuously monitoring its own performance, exposure to intrusion, vulnerability to failure or degradation, and implementing remedies spontaneously as needs arise.

A third and much higher level of representation is the ability of a system to express to another system its nature, interests and capabilities. What is implied here is not simply an indication that this is a software system written in the Java computer language, supporting the following interface protocols, and listing explicitly defined capabilities. This kind of explicit introduction is similar to the directed search capabilities that are offered by the query facilities of any database management system available today. To fully support the requirements of 'discovery' the system should be able to communicate its nature, interests and capabilities in a conceptual manner. The analogy in the database domain is a conceptual search capability, where the target of the search is only vaguely defined as being something like something else and is expected to extend beyond the boundaries of any particular database or database management system (Pohl et al. 1999, 69-74). The ability to represent this kind of 'self-awareness' in an ontology appears to be well beyond current knowledge modeling capabilities.

The Semantic Web Initiative

It is unlikely that anyone predicted in the early 1970s when the Internet first appeared on the foundations of the ARPANET project funded by the U.S. Department of Defense Advanced Research Projects Agency (DARPA) that some 30 years later in 2003 the Internet would be used on a regular basis by more than 600 million people and serve as the preferred medium for close to (US)$4 trillion in business transactions. However, although the Internet provides almost instant global connectivity and potential access to an enormous volume of information, all of that information is stored in a low-level form as data. As a result, even the most powerful search engines can do little more than pattern-match on keywords as they attempt to retrieve user requested information. The product of such data searches is typically hundreds of information source references that may or may not be useful to the human user. The latter may then have to spend hours reviewing each source to determine whether it is relevant to the purpose of the search. This was not the intention of the creators of the World Wide Web (Berners-Lee and Fischetti 1999).

There is a valid concern that the more successful the Internet becomes in providing global connectivity to millions of users, with a corresponding exponential growth in the availability of information, the less useful it will become as a source of information. Succinctly stated the evolution of the Internet, like software systems in general, has been driven by the ability of computers to rapidly manipulate vast amounts of data without any understanding of the meaning of the data being processed. The vision of the Semantic Web is intended to overcome this serious deficiency by making the information on the World Wide Web understandable by computer

software. Signs of this vision have become evident with the increasing interest in adding semantics to data.

The historical development of data manipulation and storage techniques first showed a preoccupation with efficiency, leading to the deletion of context in favor of the arrangement of data into neatly packaged records. This appeared to be a perfectly logical approach in line with the notion that the application, and not the data, is the enabler of the desired functionality. Accordingly, the data requirements were encapsulated in the application, and even when programming languages began to acquire object-oriented facilities the more prominent role assigned to data was largely hidden from the users deep inside the application.

All of this seemed to work quite well until the need for interoperability and the attendant requirement for the exchange of data among applications surfaced. Two problems were quickly recognized. First, since each application controlled its own data schema the linking of multiple applications required application-to-application data mappings that led to hardwired systems. It soon became apparent that while it was possible to maintain the vertical flow of data within each of these stovepipe systems, it was inordinately difficult to exchange data horizontally between stovepipes. The second problem centered on this need for horizontal interoperability: How to exchange data between two stovepipe systems so that the receiving application will be able to process the imported data in a useful manner? There appeared to be two possible approaches for addressing this problem. To explicitly predefine the data exchange format and content, or to add meaning-identifiers to the data. The first approach, while providing a modest level of interoperability in the short term, exacerbated the problem in the long term. The hardwired data bridges were difficult and costly to maintain, provided little (if any) flexibility, and constituted multiple system failure points. The second approach led to the definition of standard data exchange protocols that conveyed to the receiving application at least some indication of the meaning of an imported data package. Of these protocols the Extensible Markup Language (XML) is rapidly gaining widespread acceptance. XML provides a degree of syntactic interoperability through nested data record delimiters (i.e., Unicode characters), data meaning-identifiers (i.e., tags), and links to other resources (i.e., Uniform Resource Identifiers).

Does a protocol like XML convey sufficient meaning to support horizontal interoperability? The answer is, no. The XML elements that are added to a data exchange package to convey meaning are of value only if the receiving application understands the name of each element. For example, the tag name “address” is only useful to the receiving application if it interprets that name to have the same meaning as the meaning assumed by the sending application (i.e., “address” could mean street address, e-mail address, object reference ID, etc.). However, XML does provide a syntactic foundation layer on which other layers such as the Resource Description Framework (RDF) can be built. The combination of these layers will serve as the enabling structure of what is referred to as the Semantic Web.

The vision of the Semantic Web is an information-centric environment in which autonomous software services with the ability to interpret data imported from other services are able to combine their abilities to accomplish some useful intent. This intent may range from simply finding a particular item of information to the more sophisticated tasks of discovering patterns of data changes, identifying and utilizing previously unknown resources, and providing intelligent decision-assistance in complex and time-critical problem situations. An example of such an environment is the TEGRID proof-of-concept system that was first demonstrated by the Collaborative Agent Design Research Center (CADRC) during an Office of Naval Research

Workshop in Washington in September 2002 (Gollery and Pohl 2002). A brief summary of this demonstration is provided in the following section.

TEGRID: An Experimental Web Services System

The principal components of the TEGRID demonstration are ontology-based Web services that are capable of seeking and discovering existing Web services, extending their own information models through the information model of any discovered Web service, and automatically reasoning about the state of their internal information models. As shown in Figure 14, these components (referred to as Cyber-Spiders in TEGRID) consist of three principal components: a Web server; a semantic Web service; and, an information-centric application.

The Web server, utilizing the standard Hypertext Transfer Protocol (HTTP), serves as the gateway through which the Cyber-Spider gains access to other existing Web services. Existing Web servers primarily provide access to Hypertext Markup Language (HTML) data sources and perform only simple operations that enable access to externally programmed functionality. However, these simple operations currently form the building blocks of the World Wide Web.

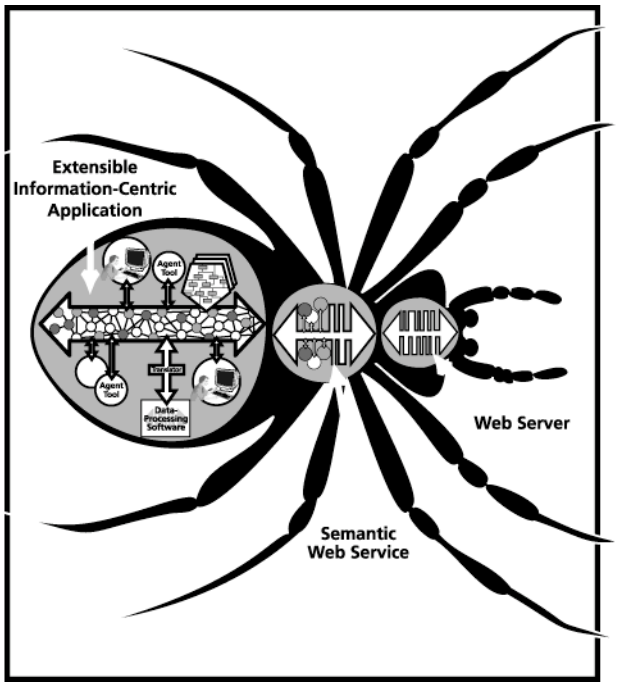


Figure14: Anatomy of a Cyber-Spider

Figure 15: Cast of TEGRID players

The second component of a Cyber-Spider is a semantic Web service (i.e., a Web service with an internal information model). A Web service is accessed through a Web server utilizing standard protocols (e.g., UDDI, SOAP, WSDL, SML) and is capable of providing programmed functionality. However, clients to a standard Web service are usually restricted to those services that implement specific predefined interfaces. The implementation of Web services in the Internet environment allows organizations to provide access to applications that accept and return complex objects. Web service standards also include a limited form of registration and discovery, which provide the ability to 'advertise' a set of services in such a way that prospective client programs can find services that meet their needs. The addition of an internal information

model in a semantic Web service allows the storage of semantic level descriptions (i.e., information) and the performance of limited operations on these semantic descriptions. In other words, the semantic Web server component of a Cyber-Spider is capable of reasoning.

The third component of a Cyber-Spider is one or more information-centric applications. These applications are designed to take advantage of the resources provided by a number of semantic Web services, enabling them to reason about the usefulness of each service as a core capability within a more sophisticated set of discovery strategies. Moreover, the application component is able to construct relationships among the information models of different services, with the ability to integrate services without requiring agreement on a common information model.

With these three components Cyber-Spiders are at least minimally equipped to operate in an Internet environment as autonomous software entities, capable of discovering needed services; accepting services from external offerers; providing services to external requesters; gaining context through an internal information model; automatically reasoning about available information; extending their information model during execution; extending their service capabilities during execution; and, learning from their collaborations.

The Cast of Players

Based on the scenario described in Figure 15, the TEGRID cast of players includes six semantic Web services: the Emergency Operations Bureau (EOB) of the Los Angeles Sheriff's Department; several Local Sheriff Stations (LSS); a Power Supply Organization (PSO); a Traffic Control Organization (TCO); several Rapid Response Teams (RRT); and, a Los Angeles County Web Services Kiosk (WSK).

Fundamental to each player are three notions. First, each player operates as an *autonomous* entity within an environment of other players. Most, but not all of the other players are also autonomous. This requires the autonomous players to be able to discover the capabilities of other players. Second, each autonomous player has a sense of *intent* to accomplish one or more objectives. Such objectives may range from the desire to achieve a goal (e.g., maintain situation awareness, coordinate the response to a time-critical situation, or undertake a predetermined course of action following the occurrence of a particular event) to the willingness to provide one or more services to other players. Third, each player (whether autonomous or not) is willing to at least *cooperate* with the other players. In some cases the level of cooperation will extend to a collaborative partnership in which the partnering players contribute to the accomplishment of a common objective. In other cases the cooperation may be limited to one player providing a service to another player, without any understanding or interest in the reason for the service request.

To operate successfully in such an autonomous Internet-based environment a Cyber-Spider player should be endowed with the following capabilities:

1. Subscribe to information from external sources (e.g., alerts, ontology extensions).
2. Accept subscriptions from external clients.
3. Dynamically change its subscription profile.
4. Extend its internal information representation.
5. Extend its own service capabilities.
6. Generate new agents for its own use.

7. Describe its own service capabilities to external clients.
8. Seek, evaluate and utilize services offered by external clients.
9. Provide services to external clients.
10. Describe its own (intent) nature to external clients.

The Cyber-Spiders in TEGRID are capable of demonstrating eight of these ten desirable capabilities. The ability of a Cyber-Spider to dynamically change its subscription profile, while technically a fairly simple matter, was not implemented because it is not used in the demonstration scenario. The ability of a Cyber-Spider to describe its own nature to external clients, on the other hand, is technically a much more difficult proposition. It will require a Cyber-Spider to have an understanding of its personality as a collective product of its internal information model and the relationship of that model with the external world. At best this must be considered a challenging research area that is beyond the current capabilities of information-centric software systems.

The Capabilities

The objective of the TEGRID scenario is to demonstrate the discovery, extensibility, collaboration, automatic reasoning, and tool creation capabilities of a distributed, just-in-time, self-configuring, collaborative multi-agent system in which a number of loosely coupled semantic Web Services associate opportunistically and cooperatively to collectively provide decision assistance in a crisis management situation. Specifically, these capabilities are defined as follows:

Discovery: Ability of an executing software entity to orient itself in a virtual cyberspace environment and discover other software services.

Extensibility: Ability of an executing software entity to extend its information model by gaining access to portions of the information model of another executing software entity.

Collaboration: Ability of several semantic Web Services to collaboratively assist each other and human users during time critical decision-making processes.

Reasoning: Ability of a software agent to automatically reason about events in near real-time under time critical conditions.

Tool Creation: Ability of a semantic Web Service to create an agent to perform specific situation monitoring and reporting functions.

The reasoning capabilities available in TEGRID are performed by software agents that are components of the players (i.e., the Cyber-Spiders). In other words, agents are predefined clients within player systems and perform internal functions that are necessary for the particular player to deliver its services and/or accomplish its intent. The following agents (i.e., collaborative tools) are available in the current TEGRID implementation:

Risk Agent: Assists the Emergency Operations Bureau to identify high-risk entities in the jurisdictional region of an activated Local Sheriff Station.

Deployment Agent: Assists the Emergency Operations Bureau to determine whether Rapid Response Team support is required for a particular activated Local Sheriff Station.

Power Level Agent: Assists the Power Supply Organization to determine if the electric power demand has exceeded supply.

Situation Agent: Assists the Emergency Operations Bureau to prepare and update its Status Report.

Station Monitor Agent: Assists the Emergency Operations Bureau to identify all Local Sheriff Stations that will experience power blackouts during the current and next blackout cycle.

Status Agent: Assists a Local Sheriff Station to prepare and update its Situation Status Report.

Local Station Agent: Assists a Local Sheriff Station to determine whether sufficient local resources are available to deal with current conditions.

Scheduling Agent: Assists the Emergency Operations Bureau to assign Rapid Response Teams and equipment to situations requiring their involvement.

Incident Agent: Assists the Emergency Operations Bureau to monitor the response to a particular situation supported by one or more of its Rapid Response Teams.

Routing Agent: Assists the Traffic Control Center to determine alternative routes to a particular situation location.

Demonstration Summary

Since the complete TEGRID demonstration scenario has been described elsewhere (Gollery and Pohl 2002) it will suffice here to summarize some typical events and automated reactions.

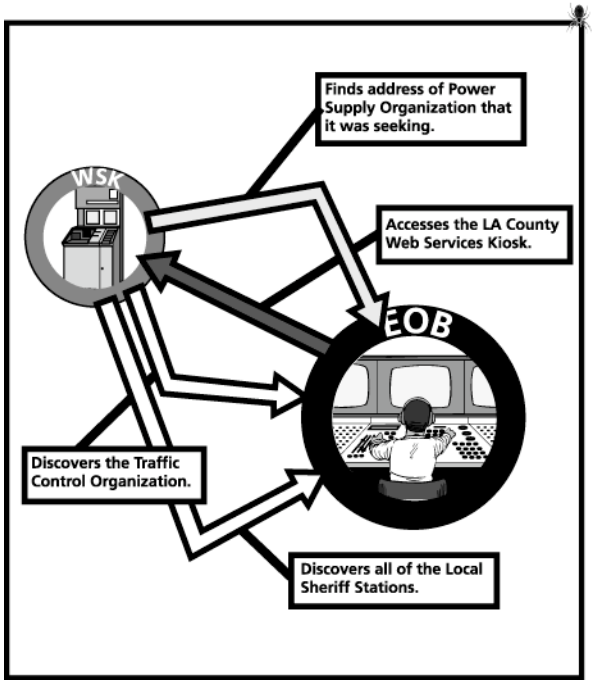


Figure 16: Orientation and discovery

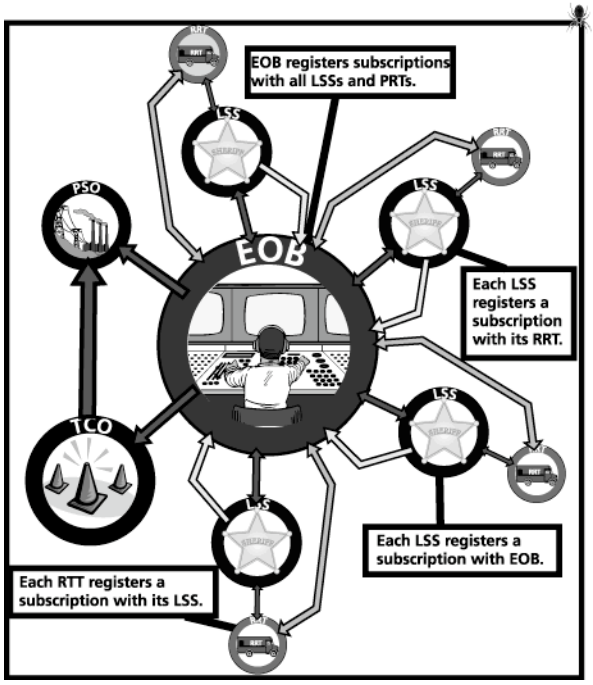


Figure 17: Information subscription

Orientation: The players orient themselves by accessing one or more directories of available services and registering an information subscription profile with those services that they believe to be related to their intent (Figure 16).

Subscription: The players access the services that they require to achieve their intent, register appropriate subscription profiles, and query for information that they believe to have a need for (Figure 17). For example, the Emergency Operations Bureau registers a subscription profile with each Local Sheriff Station, which includes all current police unit locations, mission completion events, new mission events, and any information changes relating to the availability of its Rapid Response Teams. Then queries each Local Sheriff Station for all information relating to its Rapid Response Teams and extends its information model. Finally, registers subscription profiles with each Rapid Response Team, the Power Supply Organization, and the Traffic Control Organization.

Collaboration: The Power Supply Organization first alerts its subscribers that a rolling power blackout condition is imminent (i.e., will commence per predefined schedule within 15 minutes) and subsequently alerts its subscribers that the rolling power blackout has commenced. The Emergency Operations Bureau (EOB) utilizes its Situation Agent to prepare the first version of the 'EOB Situation Status Report'. Then alerts all Local Sheriff Stations, in whose jurisdictions the next scheduled set of blackouts will occur, to prepare for potential deployment. And, finally, warns the Rapid Response Teams assigned to assist the Local Sheriff Stations in whose jurisdictions the next set of blackouts are scheduled to occur, to prepare for potential deployment. Consequently, all activated Local Sheriff Stations utilize their Status Agents to prepare the first version of their 'Situation Status Reports', the Local Sheriff Stations in whose jurisdiction the next set of blackouts is scheduled to occur, prepare for deployment.

Demonstration Results

The objectives of the TEGRID project were three-fold. First, to explore the primary capabilities that would be required of semantic Web services operating as largely autonomous decision-support components in a self-configuring, just-in-time, intelligent decision-assistance toolkit of collaborating software agents. Second, to determine if the currently available information-centric software technology could support at least basic (i.e., meaningful and useful) implementations of these required capabilities. And, third, to build a working experimental system that could serve as a test-bed for longer term research studies focused on the behavioral characteristics of self-configuring intelligent systems in general, and the ability of such systems to deal with specific kinds of dynamic and complex problem situations.

The demonstration showed that, today at a base level of functionality and in the near future at a much more sophisticated level, a Semantic Web environment will be able to support semantic Web services with the ability to: discover desired existing external services; accept and utilize services from external offerers; provide services to external requesters; gain understanding through the context provided by an internal information model; automatically reason about available information within the context of the internal information model; extend the internal information model during execution; spontaneously generate new agents during execution as the need for new capabilities arises; and, learn from the collaborations that occur within the cyberspace environment.

References

Berners-Lee T. and M. Fischetti (1999); 'Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by its Inventor'; Harper, San Francisco, California.

Cass S. (2004); 'A Fountain of Knowledge'; IEEE Spectrum (www.spectrum.ieee), Jan.30.

Dreyfuss H. (1979); 'What Computers Can't Do: The Limits of Artificial Intelligence'; Harper and Rowe, New York, New York.

Deyfuss H. and S. Dreyfuss (1986); ' Mind Over Machine: The Power of Human Intuitive Expertise in the Era of the Computer'; Free Press, New York, New York.

Dreyfuss H. (1997); 'What Computers Still Can't Do: A Critique of Artificial Reason'; MIT Press, Cambridge, Massachusetts.

Fellbaum C. (1998); 'WordNet, An Electronic Lexical Database'; MIT Press, Cambridge, Massachusetts.

Ganek A. and T. Corbi (2003); 'The Dawning of the Autonomic Computing Era'; IBM Systems Journal, 42(1) (pp.5-18).

Gollery S. and J. Pohl (2002); 'The TEGRID Semantic Web Application: A Demonstration System with Discovery, Reasoning and Learning Capabilities'; Office of Naval Research (ONR) Workshop Series on Collaborative Decision-Support Systems, hosted by the Collaborative Agent Design Research Center (CADRC) of Cal Poly (San Luis Obispo) in Quantico, VA, September 18-19.

Horn P. (2001); 'Autonomic Computing: IBM's Perspective on the State of Information Technology'; IBM Corporation, October 15 (www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf).

Lucas J. (1961); 'Minds, Machines and Goedel'; Philosophy, 36 (pp. 120-4).

Patterson D., A. Brown, P. Broadwell, G. Candea, M. Chen, J. Cutler, P. Enriquez, A. Fox, E. Kiziman, M. Merzbacher, D. Oppenheimer, N. Sastry, W. Tetzlaff, J. Traupman and N. Treuhaft (2002); 'Recovery-Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies'; UC Berkeley, Computer Science Technical Report (UCB//CSD-02-1175), University of California, Berkeley, California, March 15, 2002.

Pedersen T. and R. Bruce (1998); 'Knowledge Lean Word-Sense Disambiguitization'; Proceedings 5th National Conference on Artificial Intelligence, July, Madison Wisconsin.

Pohl J. (1998); 'The Future of Computing: Cyberspace'; in Pohl J. (ed.) Advances in Collaborative Decision-Support Systems for Design, Planning, and Execution, focus symposium: International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, August 17-21 (pp.9-28).

Pohl J., A. Chapman, K. Pohl, J. Primrose and A. Wozniak (1999); '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, 1997, reprinted July 1999 (pp.69-74).

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. (2nd Edition)

Random (1999); 'Random House Webster's College Dictionary'; Random House, New York, New York.

Searle J. (1980); 'Mind, Brains and Programs'; The Behavioral and Brain Sciences, 3 (pp. 417-24).

Searle J. (1992); 'The Rediscovery of the Mind'; MIT Press, Cambridge, Massachusetts.

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

Semantic Web Bibliography

Berners-Lee T. (2002); 'Weaving the Web'; Harper, San Francisco, California.

Berners-Lee T. (2004); 'What the Semantic Web Can Represent';
(www.w3.org/DesignIssues/RDFnot.html)

Berners-Lee T., J. Hendler and O. Lassila (2001); 'The Semantic Web'; Scientific American, May (www.scientificamerican.com/2001/0501issue/0501berners-lee.html)

Brickley D. and R. Guha (eds.) (2002); 'RDF Vocabulary Description Language 1.0: RDF Schema'; W3C Working Draft, April 30 (www.w3.org/TR/rdf-schema/)

Business Week (2002); 'The Web Weaver Looks Forward'; Interview with TIM Berners-Lee, March 27 (www.businessweek.com/bwdaily/dnflash/mar2002/nf20020327_4579.htm)

Carroll J. and J. De Roo (eds.) (2002); 'Web Ontology Language (OWL) Test Cases'; W3C Working Draft, October 24 (www.w3.org/TR/2002/WD-owl-test-20021024/)

Casey M. and M. Austin (2001); 'Semantic Web Methodologies for Spatial Decision Support'; Institute for Systems Research and Department of Civil and Environmental Engineering, University of Maryland, November.

Cohen P., R. Schrag, E. Jones, A. Pease, A. Lin, B. Starr, D. Easter, D. Gunning and M. Burke (1998); 'The DARPA High Performance Knowledge Bases Project'; Artificial Intelligence Magazine 19(4) (pp.25-49) (reliant.teknowledge.com/HPKB/Publications/AImag.pdf)

DAML-ONT (2000); (www.daml.org/2000/10/daml-ont.html)

DAML+OIL (2001); (www.daml.org/2001/03/reference.html)

DAML-S (2002); (www.daml.org/services/daml-s/0.7/)

Dean M., D. Connolly, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, F. Stein and L. Stein ((eds.) (2002); 'OWL Web Ontology Language 1.0 Reference'; W3C Working Draft 29, July and November 12 (www.w3.org/TR/owl-ref/)

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, Indiana.

Ewalt D. (2002); 'The Next Web'; Information Week, October
(www.informationweek.com/story/IWK20021010S0016)

Fikes R. and D. McGuinness (2001); 'An Axiomatic Semantics for RDF, RDF Schema and DAML+OIL'; KSL Technical Report (KSL-01-01), October
(www.ksl.stanford.edu/people/dlm/daml-semantics/abstract-axiomatic-semantics.html)

Garshol L. and G. Moore (eds.) (2002); 'The XML Topic Maps (XTM) Syntax'; JTC1/SC34:ISO 13250, July 22 (www.y12.doe.gov/sgml/sc34/document/0328.htm)

Gil Y. and V. Ratnakar (2002); 'Markup Languages: Comparison and Examples'; Information Sciences Institute, University of Southern California, TRELLIS project

(www.isi.edu/expect/web/semanticweb/comparison.html)

Heflin J., R. Volz and J. Dale (eds.) (2002); 'Requirements for a Web Ontology Language'; W3C Working Draft, July 8 (www.w3.org/TR/webont-req)
Hendler J., T. Berners-Lee and E. Miller (2002); 'Integrating Applications on the Semantic Web'; Journal of the Institute of Electrical Engineers of Japan, 122(10), October (pp.676-680).

Horrocks I. (2002); 'DAML+OIL: A Description Language for the Semantic Web'; IEEE Intelligent Systems, Trends and Controversies.

Manola F. and E. Miller (eds.) (2002); 'RDF Primer'; W3C Working Draft, March 19 (www.w3.org/TR/2002/WD-rdf-primer-20020319/)

OIL (2004); (www.ontoknowledge.org/oil/)

Ontolingua (2004); (www.ksl.stanford.edu/software/ontolingua/)

OWL (2001); 'The Web Ontology Language'; (www.w3.org/2001/sw/WebOnt/)

Patel-Schneider P., I. Horrocks, P. Payes and F, van Harmelen (eds.) (2002); 'Web Ontology Language (OWL) Abstract Syntax and Semantics'; W3C Working Draft, November 8
(www.w3.org/TR/2002/WD-owl-semantics-20021108/)

Swartz A. (2002); 'The Semantic Web in Breadth'; (logicerror.com/semanticWeb-long)

W3C (1999); 'Resource Description Framework (RDF) Model and Syntax Specification'; W3C Recommendation, February 22.

W3C (2001); 'XML Linking Language (Xlink) Version 1.0'; W3C Recommendation, June 27
(www.w3.org/TR/xlink/)

W3C (2003); 'Design Issues'; (www.w3.org/DesignIssues/diagrams/sw-stack-2002.png)

Approaching Semantic Interoperability:
A Multi-Agent Observation-Based Communication Framework

Adam Gray
ARES Agent Team Leader
CDM Technologies, Inc
San Luis Obispo, CA
adgray@cdmtech.com

Abstract

Multi-agent architectures, abstractions in which multiple autonomous processes collaborate to solve a problem, are an attractive approach to dealing with complex problems that require multiple factors to be considered simultaneously. This approach can be even more effective when humans are treated as additional agents that provide supplemental knowledge and common sense. However, such systems face considerable obstacles in providing a communication framework that allows interoperability between agents, both human and software, that possess ontologies representing significantly different views of the world. The problem is compounded further by the need to share information among agents that is often uncertain, incomplete and may even be conflicting. Current agent communication languages, such as Knowledge Query and Manipulation Language (KQML) and Foundation for Intelligent Physical Agents - Agent Communication Language (FIPA-ACL), support standardized syntactic interoperability among agents without restricting the language or ontology used. However, there is a need for a framework that promotes semantic interoperability, particularly in the context of handling uncertain communication and establishing the ontology overlap necessary for mutual agent understanding. This paper utilizes the design and implementation of the Mission Readiness Assessment System, a multi-agent decision-support system developed for the US Navy and sponsored by the Office of Naval Research, to explore the use of an observation-based communication framework that addresses semantic interoperability among agents.

Keywords

multi-agent system; decision-support system; observation-based communication; agent communication language; ontology

Introduction

The ability of the human mind to solve large, complex problems is often insufficient compared to that required for effective problem solving. The main difficulties in this regard revolve around the idea of bounded rationality, the theory of which expounds that decision makers exhibit rationality over only a subset of a problem space. Many problems appear overly difficult either because the problem-solver cannot contemplate all aspects of the problem at once or because the problem-solver lacks expertise necessary to address certain aspects of the problem. (Bird and Kasper 1996) Furthermore, problems in which decisions in a single area tend to influence many others, and therefore require multiple factors to be considered simultaneously, conflict with the sequential-based

human cognitive process. (Myers and Pohl 1995) Hence, the rationally bounded nature of the human mind leads to the limiting of human problem-solvers to a narrow domain, often restricting classes of problems in which no individual aspect is particularly complex. These limitations have generally been approached by human problem-solvers through the collaboration of multiple people possessing expertise in diverse domains. Numerous examples of these collaborative teams exist with real-world examples such as committees, boards and even software development teams. (Myers and Pohl 1995)

Multi-agent systems are an emerging software-building paradigm that attempts to deal with inherent problem-solving difficulties by modeling the characteristics of a human problem-solving team. These systems introduce a powerful abstraction in which a software agent is an autonomous goal-directed process that is capable of performing actions, is situated in, is aware of and reacts to its environment, and cooperates with other agents to accomplish its tasks. (Finin et al. 1995) Such applications have been shown to be especially effective in dealing with problems in which multiple factors need to be considered simultaneously, different types of expertise are required and dynamic changes occur in the solution strategies. (Myers and Pohl 1995)

While, automated systems have shown the ability to outperform humans in both dealing with large amounts of information and handling concurrent problem aspects, humans still possess a significant advantage in, among other things, applying common sense to unexpected situations and developing dynamic solution strategies. Thus, the multi-agent abstraction can become even more powerful by treating humans as additional agents capable of providing common sense and guidance to the software agents. (Myers and Pohl 1995) Systems in which this strategy is employed, popularly called decision-support systems, are buoyed further by the fact that human decision makers are much more likely to accept conclusions made by a software system if they are actively involved in the decision-making process.

In order for agents to engage in useful decision-making there must exist a representation of the concepts, and relationships between those concepts, inherent within the problem domain. These representations, popularly referred to as ontologies, facilitate an understanding of the problem domain by the agent that is required for utilization of the agent's reasoning capabilities. However, a number of difficulties exist in the establishment of this representation for multi-agent systems. Similar to how human collaborative teams need an overlap of knowledge large enough to allow sufficient interaction of ideas, collaborative agents require sufficient ontology overlap to allow mutual understanding. Unfortunately, in direct opposition to this need is the desire to keep agents sufficiently independent to allow unique contribution. Furthermore, large overlap between agent ontologies requires a greater number of domains be encompassed by an individual ontology, tending to result in ontologies comprised of generalities. These generalities lead to domain representations that, while partially acceptable to all, are not ideal for any individual agent. (Pohl K. 2001) Thus it becomes extremely important for an agent communication framework to allow, and simplify the process of determining, the ontological overlap sufficient to support the required collaboration among agents.

It is also important to note that in any but the simplest of examples, human collaborative teams are required to make assertions without complete information and with less than one hundred percent accuracy. Not surprisingly, computer systems that model this collaboration must deal with similar constraints. In fact, agents within a multi-agent system must not only deal with communication that can be uncertain, incomplete or even conflicting, but a framework must exist that allows for the combining of redundant observations and for the determination of their relative reliability.

Current Agent Communication Languages

Much of the research in agent communication languages (ACLs) has revolved around the idea of speech act theory, in which not just the information content is communicated but also the attitude behind that content (i.e. assertion, request, query, etc…). (Finin et al. 1999) This paper focuses on two of the most well known of these ACLs, KQML and its descendent FIPA-ACL. These languages possess a syntax that allows information passing between agents entirely independent of both the content's language used and the underlying ontology for the content.

KQML was conceived as both a message format and a message-handling protocol to support runtime knowledge sharing among agents. The key features of KQML can be summarized as follows:

- KQML messages are opaque to the content they carry and do not merely communicate sentences in some language, but rather communicate an attitude about the content.
- The language's primitives are called performatives and, as the term suggests, the concept is related to speech act theory. These performatives define the permissible actions that agents may attempt in communicating with each other. (Finin et al. 1995)

The syntax of KQML is based on a balanced parenthesis list. The initial element of the list is the performative and the remaining elements are keyword/value pairs. Figure 1 shows a KQML message from agent *joe* that represents a query about the price share of IBM stock and the resulting response from the stock-server agent. As can be seen in this exchange the content structure, language and ontology are entirely independent of the KQML language.

The Foundation for Intelligent Physical Agents (FIPA) is a non-profit association registered in Geneva, Switzerland. Its stated purpose is to promote the success of emerging agent-based applications, services and equipments. The FIPA-ACL is the agent communication language created by FIPA for systems that incorporate the FIPA standard. (Suguri 1999) The syntax of this language is based closely on KQML and the differences between the two are insignificant for the purposes of this paper.

The main advantage of both KQML and FIPA-ACL is that they promote standardized agent communication in the least restrictive way possible. The ontology and content

language used in this communication is immaterial to the syntax. The use of performatives to express the attitude behind communication also allows a degree of understanding between agents not easily supported with content alone.

```
(ask-one
:sender joe
:content (PRICE IBM ?price)
:receiver stock-server
:reply-with ibm-stock
:language LPROLOG
:ontology NYSE-TICKS)

(tell
:sender stock-server
:receiver joe
:in-reply-to ibm-stock
:language LPROLOG
:ontology NYSE-TICKS)
```

Figure 1 - Example KQML Agent Exchange

By putting no restrictions on the content language or content ontology used in agent communication, these languages provide syntactic interoperability between agents; however, they do not address the larger problem of semantic interoperability. This includes the need for a framework that helps in the handling of uncertain communication and the establishing of the ontological overlap necessary for mutual agent understanding.

Observation-Based Communication Framework

This paper proposes a framework for addressing vital aspects of semantic operability between agents. It is based on the Integrated Cooperative Decision Model (ICDM) development framework which consists of an underlying architecture, fundamental design criteria, and development tools and processes for creating agent-based decision-support systems. (Pohl 2002) The underlying architecture provides a set of high-level application-independent subsystems and the mechanisms to support collaborative interaction among them. These generic subsystems can be quickly tailored to produce an application specific architecture and implementation utilizing the ICDM development tools. The initial development of ICDM was undertaken by the Collaborative Agent Design Research Center (CADRC) at California Polytechnic State University, San Luis Obispo. Based on a three-tier architecture, ICDM incorporates technologies, such as distributed-object servers and inference engines, to provide a collaborative environment for agent-based decision-support systems that provides both developmental efficiency and architectural extensibility.

Observation-Based Communication

The proposed framework employs observation-based inference, a paradigm that entails the representation of the majority of operational information, including all agent

communication, within the system as observations. This greatly reduces the required ontology overlap and simplifies the information passing process. Each observation has a knowledge-level concept and the relationships between these concepts form the basis for agent inference. Agent rules look for generic patterns between concepts and infer new observations based on logical (and, or) patterns between them. This allows users of the system, or agents within the system, to dynamically add concepts and concept relationships. The ability for an agent to dynamically modify its own knowledge structure also provides an important foundation for learning. (Gray 2003)

The observation ontology fragment shown in Figure 2 is a simplified version of the only ontology piece that agents are required to share in order to communicate within the framework. It implements the knowledge level approach to developing intelligent information systems utilizing an abstract, domain independent, statically compiled ontology divided into two distinct levels. The operational level provides classes to serve as templates for creating object instances that record the day-to-day events within the domain. The knowledge level provides classes to serve as templates for creating object instances to record domain specific concepts, their relationships and knowledge of their application. This approach provides support for the powerful modeling concepts of dynamic and multiple classification and allows for the development of generic statically compiled ontologies that can be reused across multiple disparate domains. (Zang 2003)

The observation ontology fragment possesses a creator attribute specifying the human or software agent who made the observation along with a degree of belief attribute specifying the agent's level of confidence in that observation. The degree of belief attribute is crucial both for supporting uncertain communication as well as allowing probabilistic inferences of new observations. An evidence association is also included to allow specification of the past observations used by the agent in inferring a new one. The individual degrees of belief of an inferred observation's evidential observations is crucial for the agent in determining the degree of belief it assigns to the inferred observation.

An observation also includes a postedStartTime attribute that is critical for both the observation's temporal nature and the fact observations cannot be modified after conception. This allows a complete record of the history of agent communication and allows straightforward exploration of an agent's past states of belief. The ability to preserve the integrity of past observations, despite changes to the observational state of the system, is achieved by supporting the creation of new observations about old ones. For example, if an agent within a medical system makes an observation with a very high degree of belief that a person has a particular disease, and then a test is completed that contradicts this assertion, the agent simply makes a new observation on the old one stating that it is not valid. Observations that led the agent to this determination of validity are then associated as evidence to the new observation.

The Blackboard Pattern

The Blackboard Pattern best describes the top-level underpinnings of the ICDM architecture, and consequently the agent communication framework. This classical architectural pattern has been employed by the artificial intelligence (AI) community

since the early 1970's as an approach to problems for which no deterministic solution strategies are known. The name blackboard was chosen because the approach parallels the situation in which human experts sit in front of a real blackboard and work together to solve a problem. (Zang 2001) As this approach is very similar to that used by human collaborative teams, it is a very attractive architecture for agent-based systems that strive to model this same sort of collaboration.

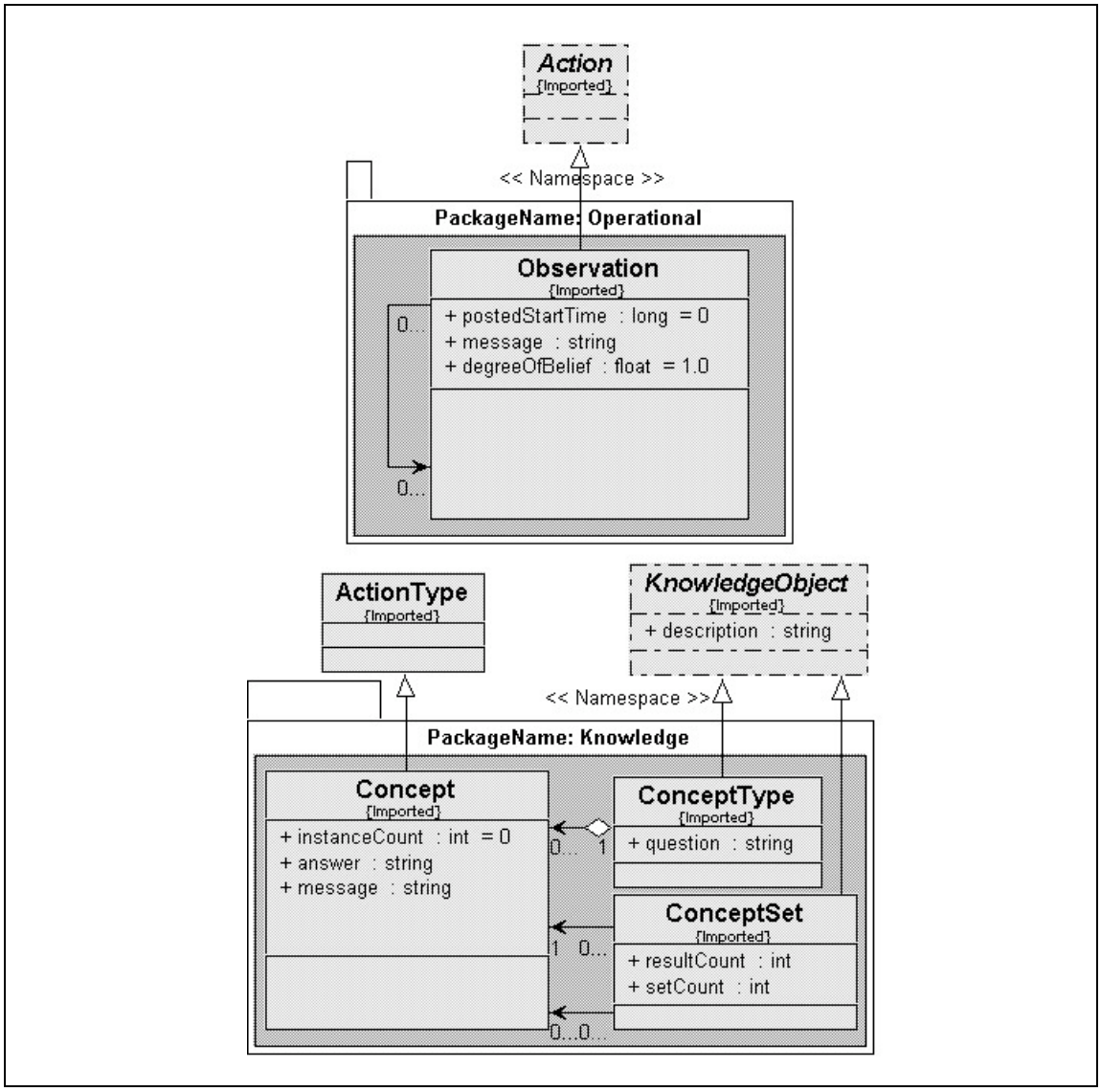


Figure 2 – Operational and Knowledge Observation Ontology Fragments

The Blackboard architecture, shown in Figure 3, employs a collection of independent programs, or agents, that work cooperatively on a common data structure, or blackboard. Each program is specialized for solving a particular part of the overall task, and all programs work together on the solution. The specialized programs are completely independent of each other and do not interact directly; this removes the many complexities inherent in one-on-one agent conversations. An observation made by an

agent is immediately accessible by all other agents who understand the context of that observation. This greatly reduces the size of ontology overlap required for agent communication as, rather then needing to share an entire ontological fragment as in the FIPA-ACL, agents only need include ontological fragments useful to their decision process.

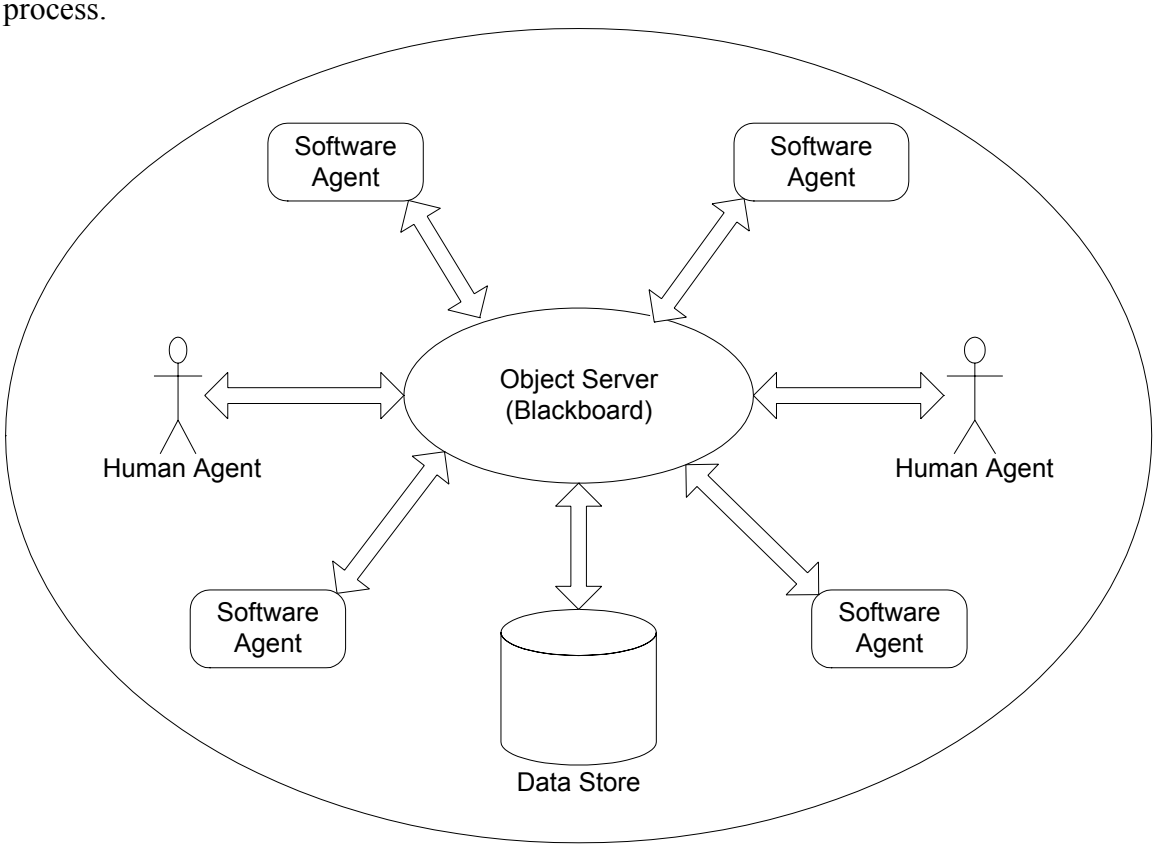


Figure 3 - Blackboard Architecture

Bayesian Networks

The proposed communication framework uses Bayesian networks to represent probabilistic relationships between concepts. This allows an agent to make probability-based inferences of new observations using uncertain observations made by other agents. A Bayesian network, or belief network, is a causal graph, associated with an underlying distribution of probability (Russel and Norvig 2003). Each leaf node within the graph contains a prior probability table and all other nodes contain a conditional probability table relating it to connected nodes (Figure 4). This representation expresses all the information contained within a joint probability distribution in a much more concise format. The inherent advantage of these graphs is that the probability of any given output variable can be determined without knowledge of all input variables.

An agent's concept hierarchy is modeled as a Bayesian network through a Concept object's association to a ConceptType and a ConceptSet (Figure 2). The ConceptType object holds the question a Concept answers and the Concept object contains an answer string specifying what the answer to that question is. For example, if a given Concept specifies that a patient has the flu, then its ConceptType's question would be "Does the

patient have the flu?" and the Concept's answer would be "yes". The Concept object is also associated to a number of ConceptSet objects as either a source or a result. These associations allow creation of a Bayesian network structure in which the probability of each result's truth is specified by the probabilities of truth of its supporting concepts. The InstanceCount attribute specifies how many times in the past an observation has been made for that Concept. Thus the Bayesian network probabilities can be dynamically improved by updating a Concept's InstanceCount attribute to coincide with the system's state. This dynamic recalculation of probabilities allows an intuitive and computationally sound vision of learning. (Gray 2003) Another advantage of using Bayesian network technology is that past operational data can be more easily incorporated into the knowledge base as the network probabilities can be determined by a combination of prior data and/or human prediction.

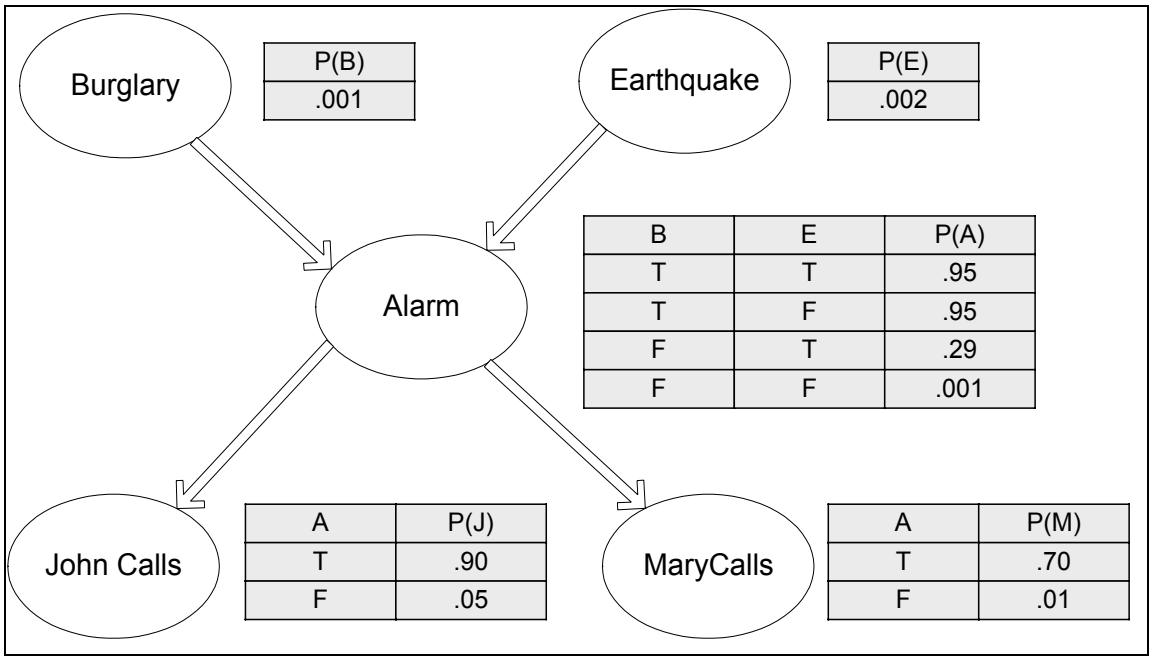


Figure 4 - A typical belief network with conditional probabilities.

Implementing Bayesian networks for use by agents within such systems has some inherent difficulties. A conditional probability table is difficult to model in an object-based format without necessitating a large number of objects. This is especially the case in belief networks with probability nodes influenced by more than two other nodes or that contain probability nodes with more than two variables. The number of objects necessary to represent a Bayesian network is on the order of $\Theta(n * v^k)$ where n is the number of nodes, v is the number of variables per node and k is the number of nodes that directly influence each other node. As there is an exponential relationship between the number of necessary objects and the number of nodes directly influencing each other node, the number of objects can become large very quickly. However, this representation is significantly more efficient than the use of a full joint probability distribution. Consider a network with 20 nodes $(n = 20)$, 5 parents per node $(k = 5)$ and 2 Boolean variables

$(v = 2)$, the Bayesian network requires approximately 640 nodes while the full joint probability distribution requires over a million nodes. (Gray 2003)

Case-Based Reasoning

The employment of humans as additional agents within the decision-support paradigm introduces a number of complexities. At the forefront is the need to allow communication between software agents and human domain experts whose representations of the world are significantly different. The proposed agent communication framework employs Taxanomic Case-Based Reasoning System (TCRS) v1.0, a taxonomical conversational case-based reasoning system developed at the Naval Research Laboratory. (Aha and Gupta 2002) TCRS supports problem solving by recalling and applying past experiences, or cases, that are similar to the problem at hand. This allows a user to incrementally specify a query by providing text annotations and answering prompted questions. This query is combined with the answers to relevant questions and the result is matched against previous cases to determine the most similar past experience. Therefore, this tool allows a software agent to determine the objectified phenomenon most similar to that which a domain expert wishes to express by analyzing the domain expert's textual description and his answers to a number of relevant questions.

The similarity measure used by TCRS to determine the correlation between two text strings is calculated using the different trigrams present in the two strings. This calculation can be seen in Figure 5 where *tri(x)* is the set of trigrams in x. For example, *tri(eloquent) = {elo, loq, oqu, que, ent}*.

$$sim(x, y) = 1 \Big/ 1 + |tri(x)| + |tri(y)| - 2 * |tri(x) \cap tri(y)|$$

Figure 5 - Similarity measure between two text strings x and y

TCRS also features taxonomical relationships between past cases to help handle abstraction difficulties inherent in phenomenon correlation. For example, the phenomenon displayed in Figure 6 would be represented in a hierarchical structure of cases within TCRS. This makes the case representation more efficient as it is indexed by fewer and only the most specific question-answer pairs available at the time of indexing. It also eliminates any unwanted correlation among features that could result from inherent abstraction. Lastly, it makes the conversation responsive to the level of abstraction in a user's query. This allows a correlation between the user's level of expertise in a particular domain and the level of detail in a case hierarchy.

(1.) The weather was bad
  (1.1) The weather was stormy
    (1.1.1) The wind speed was very high
      (1.1.1.1) The wind speed was over 90 mi./hr

Figure 6 - Phenomenon Taxonomy

Conflict Resolution Agent

The framework employs an autonomous, distinct Conflict Resolution Agent to assist in dealing with observations that are conflicting, supporting or redundant. When observations are found to be conflicting, the agent uses past reliability statistics on the observations' sources, the observations' relevance to the problem and the observations' times of conception to determine the one that it believes to be the most reliable. A 'No longer valid' observation is then posted by the Conflict Resolution Agent on the observations that were not chosen. By implementing conflict resolution in this fashion the option is afforded to all agents to either take the advice of the Conflict Resolution Agent or act on their own accord. Reliability statistics used in this analysis can be updated dynamically as determinations are made about an observation's validity, allowing the combination of real-time and historical data to improve the conflict resolution performance.

The Conflict Resolution Agent deals with supporting observations by inferring a single top-level observation with a combined degree of belief. The supporting observations used to create the new observation are associated as evidence. This reduces the number of observations an agent must process without altering or removing existing information.

Similarly, the agent deals with redundant observations by creating a top-level observation tying those observations together. The new observation is given a degree of belief befitting the combined information. This provides straightforward notification to other agents that they need not process each of the redundant observations without restricting their ability to access them.

Mission Readiness Assessment System

The Mission Readiness Assessment System (MRAS) is an idea developed by Dr. Philip Abraham at the Office of Naval Research and manifested into a funded project as an analysis of logistic related decisions afloat and a demonstration of decision support systems and agent based software for enhancing ship readiness. It represents a decision aid for a ship's commanding officer and senior enlisted personnel. Specifically, the system:

- Provides agent based shipboard decision support to the commanding officer and senior enlisted personnel.
- Allows shipboard users to view and develop the operational schedule and relate tasks to their necessary resources.
- Provides shipboard users with agent generated alerts and change notifications in response to changes in the readiness status.
- Allows shipboard users to customize and extend status reporting features and mechanisms.
- Integrates with existing shipboard information, control, and monitoring systems.

Top-level Mission Readiness assessments are presented with symbols providing immediate visibility of a ships readiness status biased by mission type. The possible status levels include:

- ***Fully Mission Capable*** *- All major equipment and systems are fully capable of performing all required functions without reservations.*

- ***Mission Capable*** *– All major equipment and systems are capable of performing all required functions with some reservations.*

- ***Marginally Mission Capable*** *– All major equipment and systems are capable of performing all required functions with major reservations.*

- ***Not Mission Capable in Selected Areas*** *– Capable of performing selected major functions in a primary mission area.*

- ***Not Mission Capable*** *– Major discrepancies exist in one or more key functional areas, making the ship incapable of accomplishing a primary mission.*

The user interface for MRAS can be seen in Figure 7. The top panel displays the current readiness of the ship for such distinct missions as: Amphibious Warfare, Medical Support and Self Defense. The right-hand side panels shows varying levels of the equipment hierarchy for each of the above missions. The left-hand side panel displays agents employed by the system including: Mission Readiness Assessment Agent, Interface Agent, Personnel Agent, Combat Systems Agent and Supply Agent. These agent icons are highlighted with yellow if they contain a current warning for the user and red if they contain a violation. The middle panel shows a graphical display of the ship with problem areas highlighted in red.

The observation-based communication framework detailed in this paper is used extensively within MRAS to support communication between agents, both human and software. Software agents within the system use the observation ontology fragment and all information passing between those agents is accomplished through the posting of observations to the blackboard. The two most prominent agents are the Equipment Status Agent, which processes inputs from outside sources, translates them to equipment level assessments and propagates those assessments up the equipment hierarchy, and the Mission Readiness Assessment Agent which infers mission level assessments from thes status of relevant equipment. User observations can be made through the use of the case-based reasoning tool TCRS and the Conflict Resolution Agent is used to handle conflicting, supporting and redundant observations.

The concept hierarchy used within MRAS varies depending on the class of the target ship, however the basic structure consists of the phenomenon that affect the ship's overall mission readiness and are associated using the Bayesian network structure detailed earlier. Use of this structure, along with the ability to specify an observation's degree of belief, allows the utilization of uncertain observations and the probabilistic inference of additional observations. This is particularly important within MRAS as situations are often faced in which determination of the exact problem causing equipment degradation is not possible.

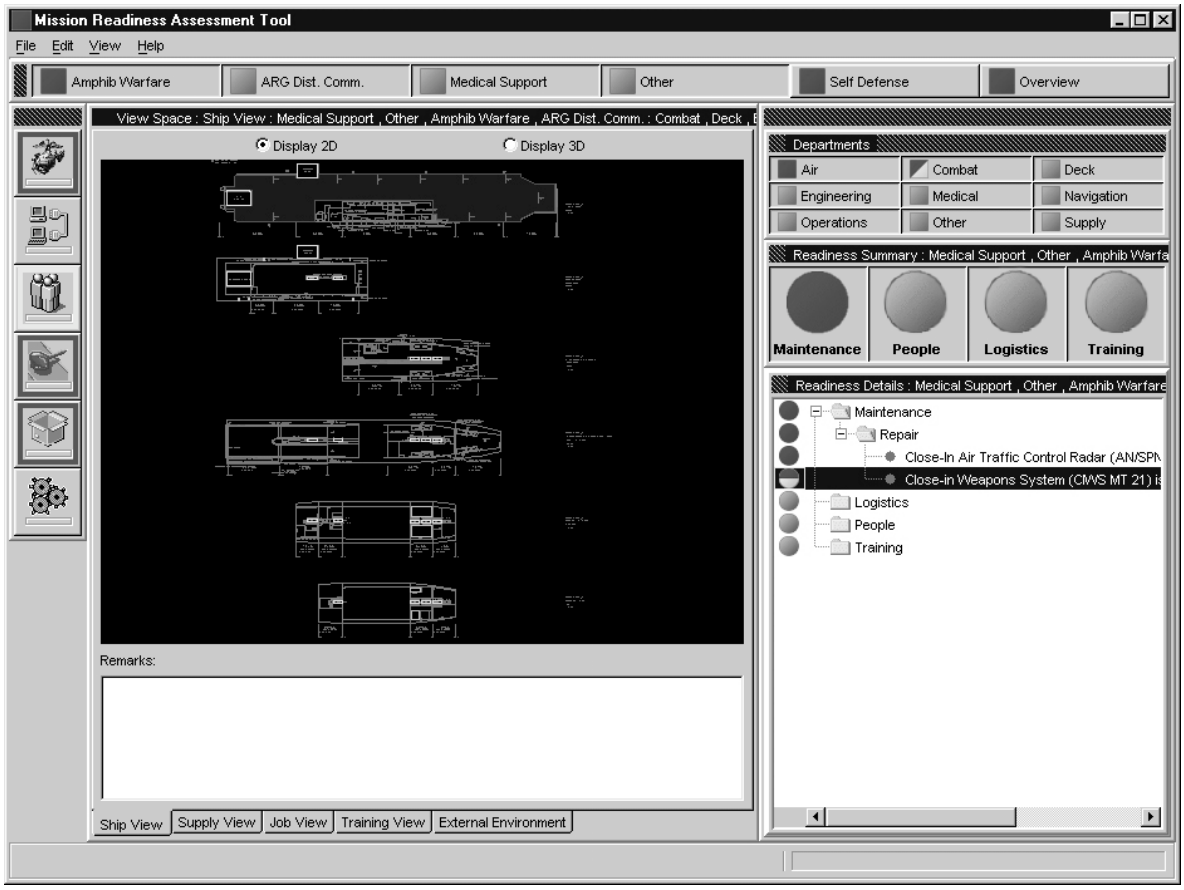


Figure 7 - SILS MRAT User Interface

TCRS is employed by MRAS to allow a user to express a phenomenon about a specific ship subsystem to the software agents. As it is not feasible for the user to search through a list of all possible phenomenon understood by the software agents, the tool allows the user to communicate by entering a textual description of the phenomenon. TCRS then presents the user a ranked list of the phenomenon it believes are most similar to the provided description along with a list of questions it believes will help improve the accuracy of that ranking. The user may choose the most similar phenomenon from the list or answer any number of the presented questions to further improve the list's accuracy.

The Conflict Resolution Agent also plays a vital role within MRAS. Assessment values are given to equipment on the ship and these values are propagated by MRAS up the tree to the mission level. As assessments are often made by multiple agents on overlapping pieces of the tree, conflicting assessment values for the same node are common. The Conflict Resolution Agent uses source reliability statistics, relevance and the observation's time of conception to determine which of the conflicting assessments to accept.

The majority of difficulties observed in the utilization of the proposed agent communication framework within the context of MRAS revolve around performance.

Bayesian network technologies have existed for many years, yet few fielded systems using the technology exist. This is due primarily to the large number of objects required for modeling any but the simplest of belief networks. As computing power has increased, Bayesian network technology has become increasingly more viable, yet, given the size of the network required for MRAS to be fielded on a naval vessel, the use of the technology is still a significant bottleneck.

The use of static observations for communication has also hindered the overall performance of MRAS. As observations within MRAS do not change after their initial inception, the number of observations within the system can quickly reach very large numbers leaving a considerable memory footprint. While partitioning the system's operational data between multiple sessions has helped reduce this overhead it is still a noteworthy constraint.

Conclusion

In recent years, multi-agent architectures have proved to be a powerful abstraction for modeling the decision making process of a human collaborative team. Such applications are especially effective in dealing with problems in which multiple factors need to be considered simultaneously and multiple types of solution strategies need to be employed. The use of humans as additional agents capable of providing commons sense and overarching authority has further increased their effectiveness. However, a number of difficulties exist in the implementation of these systems. While the differing viewpoints of collaborating agents is one of the major strengths of multi-agent systems, these differing viewpoints are also one of the most significant impedances in allowing agent collaboration. This is due to not only to the difficulty of determining the ideal agent ontology overlap but also to the need to allow uncertain communication between the agents.

This paper has proposed an observation-based multi-agent communication framework to deal with the semantic interoperability aspects of agent communication not addressed by current agent communication languages. The framework uses observation-based communication to help deal with the determination of the ideal ontology overlap and uses Bayesian networks, case-based reasoning and conflict resolution technologies to support uncertain communication.

The Mission Readiness Assessment System, a decision-support system sponsored by the Office of Naval Research, implements the proposed agent communication framework. Its implementation has allowed the expression of uncertainty in ship system assessments as well as the representation of probabilistic relationships between assessment phenomenon. However, the system has been forced to cope with performance overhead the framework introduces through its use of Bayesian network technology and static treatment of observations. Improvements in the algorithms used as well as advancements in computing power in the upcoming years should help ease these concerns as use of the framework in a fielded system approaches fruition.

References

Aha, David W. and Gupta, Kalyan Moy (2002); "Causal Query Elaboration in Conversational Case-Based Reasoning"; Proceedings of FLAIRS 2002.

Bird, Shaun D. George M. Kasper (1996); "Modeling Belief and Preferences in Multi-Agent Systems"; Proceedings of the 29th Annual Hawaii International Conference on System Sciences 1996.

Finin, Tim. Yannis Labrou. James Mayfield (1995); "KQML as an agent communication language"; September 1995.

Finin, Tim. Yannis Labrou. Yun Peng (1999); "The Interoperability Problem: Bringing together Mobile Agents and Agent Communication Languages"; Proceedings of the 32nd Hawaii International Conference on System Sciences 1999.

Fowler, Martin (1997); "Analysis Patterns, Reusable Object Models"; Addison-Wesley Longman 1997.

Gray, Adam (2003); "ARES Agent Technology Overview"; Office of Naval Research Proceedings, 2003.

Myers, Leonard. Jens Pohl (1995); "ICDM: Integrated Cooperative Decision Making – In Practice"; 6th IEEE Conference, New Orleans, July 1995.

Nickles, Matthias (2003); "An Observation-based Approach to the Semantics of Agent Communication"; Research Report FKI-24x-03, AI/Cognition Group, Technical University of Munich 2003.

Pohl, Kym (1998); "The Round Table Model: A Web-Oriented, Agent-Based Approach To Decision-Support Applications"; InterSymp-98 Conference Proceedings. Baden-Baden, Germany, August 17-21 1998.

Pohl, Kym J (2001). "Prospective Filters as a Means for Interoperability Among Information-Centric Decision-Support Systems"; 2001.

Russell, Stuart, Peter Norvig (2003); "Artificial Intelligence A Modern Approach 2nd Edition"; Prentice Hall, Englewood Cliffs, NJ 2003.

Suguri, Hiroki (1999); "A Standardization Effort for Agent Technologies: The Foundation for Intelligent Physical Agents and Its Activities"; Proceedings of the 32nd Annual Hawaii International Conference on System Sciences. 1999.

Zang, Michael (2001); "The Architecture of a Case Based Reasoning Application"; Office of Naval Research Proceedings 2001.

Zang, Michael (2002); “Data, Information and Knowledge in the Context of SILS”; Office of Naval Research Proceedings 2002.

Zang, Michael (2003); “The Knowledge Level Approach To Intelligent Information System Design”; Office of Naval Research Proceedings 2003.

Theory of Standard K-languages as a Model of a Universal Semantic Networking Language

V.A. Fomichov

Department of Mathematical Methods and Software for Information Processing
and Control Systems, Faculty of Applied Mathematics,
Moscow State Institute of Electronics and Mathematics
(Technical University), B.Tryochsvyatitelsky 1-3/12, 109028 Moscow, Russia
E-mail: vdrfom@aha.ru

Abstract

For eliminating the existing language barrier between the users of the Internet from different countries, the Japan researchers H. Uchida and M. Zhu proposed 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. Since the end of 1990s, 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. The coordinator of this project is the UNO Institute for Advanced Studies by the Tokyo University.

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).

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

Keywords

UNL; universal semantic networking language; natural language processing; semantic representation; standard K-languages; electronic contracting; e-negotiations.

Introduction

In 1999, more than 97% of all Internet hosts were in developed countries, corresponding to only 16% of the world population. As network technologies have been improving and expanding, the Internet has begun to spread throughout many countries. As a result, the following fundamental problem has emerged: how to eliminate the language barrier between the users of the Internet in different countries.

For solving this problem, the Japan researchers H. Uchida and M. Zhu proposed 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 relationships (Uchida et all, 1999).

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 the end of 1990s, 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. The coordinator of this project is the UNO Institute for Advanced Studies by the Tokyo University. At the moment, under the framework of this project, the NLPSs for six official UNO languages are being elaborated (English, Arabic, Spanish, Chinish, Russian, and French), and also for 10 other languages, including Japanese, Italian, and German.

UNL represents sentences in the form of logical expressions, without ambiguity. These expressions are not for humans to read, but for computers. Adding UNL to the network platforms will change the existing communication landscape. 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 users. Information expressed in UNL can be converted into the user's native language with higher quality and fewer mistakes than the computer translation systems. In addition, UNL, unlike natural language, is free from ambiguities.

The conclusion is drawn 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).

This paper discusses the shortcomings of UNL and proposes a new way for developing a USNL. This way is to use the theory of standard K-languages (SK-languages) represented, in particular, in (Fomichov, 1996, 2002a, b, c) as a model of a Universal Semantic Networking Language. The examples of building semantic representations (SRs) of natural language texts (NL-texts) and of representing knowledge pertaining to medicine, biology, and business are considered.

UNL as an Initial Version of a Semantic Networking Language

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. However, 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.

Besides, 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 one of possible versions of a semantic language for the Internet, or as a version of *semantic networking language*.

In this connection it is possible to establish an analogy between the studies on the creation of a semantic networking language (one of its versions being UNL), and the researches on the development of the languages for forming Web-documents. During the 1990s, one observed the stormy growth of the World Wide Web, where for representing information one used mainly the the language of marking-up hypertexts HTML. However, the language HTML was not destined for distinguishing the meaningful parts of electronic documents; this lead to considerable difficulties from the standpoint of searching the documents being relevant to the requests of end users.

That is why in the second half of the 1990s the World Wide Web Consortum (usually denoted as W3C) started to prepare the transition to new, semantically-structured means of representing information in Web-documents. During several years, two new interconnected language systems were elaborated: a language for describing metadata about informational resources RDF (Resource Description Framework) _ the language RDF SSL (RDF Schema Specification Language). These results provided a basis for announcing the start of a large-scale project of Semantic World Wide Web (Semantic Web 2001).

Taking into account the above said, we can assume that the language UNL, broadly advertised today, is not final but only initial version of a semantic networking language. The demands of formal 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 (SRs) 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 (NL-texts) with the aim to find (if possible) a model for constructing a universal or widely applicable semantic networking language.

An Outline of the Theory of Standard K-languages

The analysis of the scientific literature on artificial intelligence theory, mathematical and computational linguistics shows that today the broadest prospects for building semantic representations (SRs) of NL-texts (i.e., for representing meanings of NL-texts in a formal way) are opened by the theory of standard K-languages (SK-languages), represented in numerous publications of the author in English and Russian, in particular, in (Fomichov 1992 – 2002c).

The theory of SK-languages is a mathematical refinement of the following discovery in linguistics: a system of such 10 operations on structured meanings (SMs) of NL-texts is found 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. As a result, the RKCL-theory is a discovery in mathematical linguistics. The formal side is stated very shortly below.

At the first step (consisting of a rather long sequence of auxiliary steps), the theory defines a class of formal objects called conceptual bases (c.b.). Each c.b. B is a system of the form ((c[1], c[2], c[3], c[4]), (c[5],..., c[8]), (c[9],..., c[15])) with the components c[1],..., c[15] being mainly finite or countable sets of symbols and distinguished elements of such sets. In particular, c[1] = St is a finite set of symbols called sorts and designating the most general considered concepts; c[2] = P is a distinguished sort "sense of proposition"; c[5] = X is a countable set of strings used as elementary blocks for building knowledge modules and semantic representations (SRs) of

texts; c[6] = V is a countable set of variables; c[8] = F is a subset of X whose elements are called functional symbols.

The component c[3] = Gen is a binary relation (a partial order) on St such than if (a,b) belongs to Gen then either a = b or a concept corresponding to a is a concretization of a concept corresponding to b. The component c[7] = tp is a mapping from the union of X and V into some countable set Tps of strings called types and characterizing elements from X and V. Assume, e.g., that X contains elements ins, ed.board, Tom.Soyer designating the sort "intelligent system", the concept "editorial board", and a concrete man. Then the values of tp for elements ed.board and Tom.Soyer may be ^{ins} and ins , respectively. For the designation of the editorial board of arbitrary concrete edition, the value of tp may be the string {ins}. So types help us to distinguish objects and concepts qualifying these objects, sets and concepts qualifying these sets.

Each c.b. B determines three classes of formulas Ls=Ls(B), Ts=Ts(B), Ys=Ys(B) (l-formulas, t-formulas, anf y-formulas). The set Ls(B) is called the standard K-language in the basis B. Its strings are convenient for building semantic representations (SRs) of NL-texts. Each formula from Trs(B) has the form d & t , where d belongs to Ls(B), t is a type from Tps(B).

The formulas from Ys(B) have the form a[1] & ... a[n] & d , where a[1], ..., a[n], d belong to Ls(B), n is not the same for various d, and d is built out of a[1],..., a[n] as out of "blocks" (some of these blocks may be slightly transformed) by applying only one time some inference rule. In order to determine for arbitrary s.c.b. B the classes of formulas Ls, Ts, Ys, a group of inference rules P[0], P[1],..., P[10] is defined. The ordered pair Ks(B) = (B, Rls), where Rls is the set consisting of all these rules, is called the K-calculus in the c.b. B.

The rule P[0] provides an initial stock of l-formulas and t-formulas. Let z belong to X(B) or V(B), t is a type from Tps(B), and tp(z)=t. Then, according to the rule P[0], z belongs to Ls(B), and the string of the form z & t belongs to Ts(B).

Let's regard (ignoring many details) the structure of l-formulas (called also K-strings) which can be obtained by applying any of the rules P[1],..., P[10] at the last step of inferencing these formulas. The rule P[1] allows us to build l-formulas of the form q c where q is a semantic item corresponding to the meanings of such words and expressions as "some", "any", "arbitrary", "each", "all", "several", "many", etc. (such semantic items will be called intensional quantifiers), and c is a designation (simple or compound) of a concept. Examples of l-formulas (K-strings) for P[1] as the last applied rule are as follows:

*certn person, certn group * (Compos1, student)(Number,12),*
*every person, every person * (Age,30.year).*

The rule P[2] is destined for constructing the strings of the form f(a[1],..., a[n]), where f is a designation of a function, n>=1, a[1],..., a[n] are l-formulas built with the help of any rules from the list P[0],..., P[10]. The examples of l-formulas built with the help of P[2]: Cities(Europe), Number(Cities(Europe)). The rule P[3] enables us to build the strings of the form (a1 = a2) , where a1 and a2 are l-formulas formed with the help of any rules from P[0],..., P[10], and a1 and a2 represent the entities being homogeneous in some sense. Examples of K-strings for P[3]:

(y1 = Tilburg), (Author(War.and.Peace) = L.Tolstoy).

The destination of the rule P[4] is, in particular, to build K-strings of the form r(a[1],..., a[n]), where r is a designation of n-ary relation, n>=1, a[1],..., a[n] are the K-strings formed with the aid of some rules from P[0],..., P[10]. The examples of K-strings for P[4]:

Belongs(Namur, Cities(Belgium)), Subset(Cities(Belgium), Cities(Europe)).

The rule P[5] allows us to construct the K-strings of the form d : v , where d is a K-string not including v, v is a variable, and some other conditions are satisfied. Using P[5], one can mark by variables in the SR of any NL-text: (a) the descriptions of diverse entities mentioned in the text (physical objects, events, concepts, etc.), (b) the SRs of sentences and of larger texts' fragments to which a reference is given in any part of a text. Examples of K-strings for P[5]: all person : Z1, Less(Age(J.Smith),30.year) : P1. The rule P[5] provides the possibility to form SRs of texts in such a manner that these SRs reflect the referential structure of NL-texts. The examples illustrating this are considered below.

The rule P[6] permits to build the K-strings of the form □ *d,* where d is a K-string satisfying a number of conditions. The examples of K-strings for P[6]:

□ *poet,* □ *Belongs(Bonn, Cities(Belgium)).*

Here □ designates the connective "not".

Using the rule P[7], one can build the K-strings of the forms *(a[1] □ a[2] □ ... □ a[n])* or (*a[1] a[2] ... a[n]),* where n>1, a[1],...., a[n] are K-strings designating the entities which are homogeneous in some sense. In particular, a[1],..., a[n] may be SRs of assertions (or propositions), descriptions of physical things, descriptions of sets consisting of things of the same kind, descriptions of concepts. The following strings are examples of K-strings (or l-formulas) for P[7]:

(Finnland Norway Sweden),
(Belongs((Namur □ Leuven □ Ghent), Cities(Belgium)) □
□ *Belongs(Bonn, Cities((Finnland Norway Sweden))))).*

The destination of the rule P[8] is to build, in particular, K-strings of the form

*c * (r[1],b[1]),..., (r[n],b[n]) ,*

where c is an informational item from the primary universe X designating a concept, for i=1,...,n, r[i] is a function with one argument or a binary relation, b[i] designates a possible value of r[i] for objects characterized by the concept c. The following expressions are examples of K-strings for P[8]:

*man * (F.name,'Peter')(Year.of.studies,1),*
*group * (Compos1, student)(Number,21), turn * (Orientation,left) .*

The rule P[9] enables to build, in particular, the K-strings of the forms #A# v (des) D and #E# v (des) D, where #A# is the universal quantifier, #E# is the existential quantifier, des and D are K-

strings, des is a designation of a prime concept ("person", "city", "integer", etc.) or of a compound concept ("integer greater than 200", etc.). D may be interpreted as a SR of an assertion with the variable v about any entity qualified by the concept des. The examples of K-strings for P[9] are as follows: #A# x1 (nat) #E# x2 (nat) Less(x1,x2), #E# y (country * (Location, Europe)) Greater(Number(Cities(y)),15).

The rule P[10] allows us to construct, in particular, the K-strings of the form *<a[1],..., a[n]>,* where *n>1, a[1],..., a[n]* are K-strings. The strings obtained with the help of P[10] at the last step of inference are interpreted as designations of n-tuples. The components of such n-tuples may be not only designations of numbers, things, but also SRs of assertions, designations of sets, concepts, etc. Using jointly P[10] and P[4], we can build the string
*Study1(<Agent1,some man * (F.name,'Peter')><Institution,*
Moscow.State.Univ.>,<Time, 1996>),
where the thematic roles Agent1, Institution, Time are explicitly represented.

The scheme set forth above gives only a very simplified impression about a thoroughly elaborated new mathematical theory including, in particular: (a) the definitions of the class of K-calculuses and the class of standard K-languages in conceptual bases; (b) a number of theorems stating some properties of the new languages; (c) the definition and investigation of the properties of a new class of partial algebras called algebraic systems of conceptual syntax and providing the definition of the class of K-languages as languages isomorphic in some strict mathematical sense to standard K-languages.

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

The analysis shows that it is not difficult to approximate all expressive mechanisms of UNL by means of SK-languages, because the rule P[4] is destined for constructing formulas with the names of n-ary relationships and the rule P[8] 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, Zhu, and Della Senta, 1999). This expression can be approximated by the K-string *S1* of the form
*Destination (certn train * (Concretization, thing), certn city * (Name, 'London')*
or by the K-string *S2* of the form
*certn train * (Concretization, thing)(Destination, certn city * (Name, 'London')).*

However, the theory of SK-languages possesses many of important advantages as concerns constructing a semantic networking language of a new generation in comparison with UNL. Let's illustrate a number of such advantages. Preliminary, let's introduce the concept of a K-representation of a NL-text. If T is a NL expression in NL and a string *E* from a SK-language can be interpreted as a semantic representation (SR) of T, then *E* will be called a K-representation (KR) of the expression T.

Example 1. 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 first-level 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, certn purpose * (Examples, (feeding migration defence)))))*

The analysis of this formula enables us to conclude that it is convenient to use for constructing semntic 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 *certn 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 2. Let T2 = "All granulocytes are polymorphonuclear; that is, they have multilobed nuclei". Then T2 may have the following KR *Expr2:*

(Property(arbitr granulocyte : x1, polymorphonuclear) : P1
Explanation(P1, If-then (Have1(<Subjectt1, x1>,<Object1, arbitr 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 P[5]; it enabled us 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 contains on 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 3. Let T3 = "Type AB blood group - persons who possess types A and B isoantigens on red blood cells and no agglutinin in plasma". Then the following formula may be interpreted as a KR of T3:

*Definition (type-AB-blood-group, certn set * (Compos1, person) : S1,*
#A# x1(person) If-and-only-if(Belong1(x1, S1), (Have1(<Subject1, x1>,
*<Object1, (certn set * (Compos1, type-A-isoantigen) □*
*certn set * (Compos1, type-B-isoantigen))>,*
*<Location, certn set * (Compos1, cell1 * (Part, certn red-blood **
(Belong2, x1))>) □ □ Have1(<Subject1, x1>, <Object1,
*certn set * (Compos1, agglutinin)>, <Location, certn set **
*(Compos1, plasma1 * (Belong2, x1))>))))* .

Here #A# is the universal quantifier, the string Compos1 designates the binary relation "Qualitative composition of a set", the string *certn* is interpreted as the referential quantifier.

Example 4 (The. possibility of constructing the compound designations of goals).
Let T4 = "The owner of an insurance police calls the firm "Europ Assist"in order to tell about a damage of a car". Then T4 may have a KR

*Situation (e1, telephone-call * (Agent1, certn person * (Owner, certn insur-police1))(Object2, certn firm1 * (Name, "Europ Assist")(Goal, info-transfer * (Theme1, certn damage * (Object1, certn 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 designations of assertions (as in predicate logic) but also

designations of concepts, objects, sets of objects; (4) reflecting the semantic structure of he 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 SK-languages 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 (Uchida, et al. 1999; Uchida and Zhu 2001; Zhu and Uchida 2002).

In (Fomichov 1996 – 2002c), the hypothesis is formulated that the theory of standard K-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 standard K-languages can be used as a model of a Universal Semantic Networking Language.

Conclusions

The analysis of expressive power of the language UNL provided 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 informational languages for forming Web-documents. The conclusion is 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 standard K-languages (SK-languages) 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 theory of SK-languages can be used as a model for the development of a Universal Semantic Networking Language.

Besides, the definition of the class of SK-language can be used for the elaboration of formal languages destined for representing the records of commercial negotiations carried out by computer intelligent agents and for forming the contracts being the results of such negotiations.

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 (Slovenia); Vol. 20, No. 1 (pp. 5-32)

Fomichov, V.A. (2002a); Mathematical Foundations of Representing Meanings of Texts for the Development of Linguistic Informational Technologies. Part 1. A Model of the System of Primary Units of Conceptual Level; Informational Technologies (in Russian), No. 10 (pp. 16-25).

Fomichov, V.A. (2002b); Mathematical Foundations of Representing Meanings of Texts for the Development of Linguistic Informational Technologies. Part 2. A System of the Rules for Building Semantic Representations of Phrases and Complicated Discourses; Informational Technologies (in Russian), No. 11 (pp. 34-45).

Fomichov, V.A. (2002c); 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: Berlin, Heidelberg, New York, Barcelona, Hong Kong, London, Milan, Paris, Tokyo (pp. 183-196).

Semantic Web (2001); Semantic Web Activity Statement. W3C, URL http://www.w3.org/2001/sw/activity.

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

Uchida H. and Zhu M. The Universal Networking Language beyond Machine Translation. UNDL Foundation. 2001.

Zhu M. and Uchida H. Universal Word and UNL Knowledge Base//Proceedings of the International Conference on Universal Knowledge Language (ICUKL-2002), 25-29 Novenmber 2002, Goa of India. http://www.unl.ias.edu/publications/UW%20and%20UNLKB.htm.

Section 2:

Management of Data in Context

Real-Time Object-Oriented Databases: Theory and Applications

Michael P. Card

Sensis Corporation
5793 Widewaters Parkway, Dewitt, NY 13214

Abstract

Embedded applications, such as intelligent sensor networks, are increasingly responsible for managing large volumes of complex data. The services of a database engine (e.g., transaction semantics, data distribution, fault tolerance and failure recovery, persistence) are becoming more and more important to these applications.

Traditional database engines are not designed for use in real-time systems where sub-millisecond transaction execution times are required, nor are they designed for use in embedded systems where long-term unattended operation and careful use of resources is required. In addition, traditional database engines are typically based on the relational data model, which requires the application programmer to map programming language objects to and from relational tables. All of these factors reduce performance and inhibit the use of database technology in the real-time and embedded problem domains.

This paper presents some of the new work being done in real-time object-oriented databases, and explores the benefits this technology offers to application developers. Examples will be provided of how this technology has been used in demanding applications like real-time pattern recognition and how it could be used in both military and commercial distributed sensor networks.

Keywords

object-oriented; database; real-time; pattern recognition; spatial data; spatial index

What is an object-oriented database?

An object-oriented database is a piece of software that extends the capabilities of an object-oriented programming language to include object persistence and transaction semantics. This is shown in Figure 1.

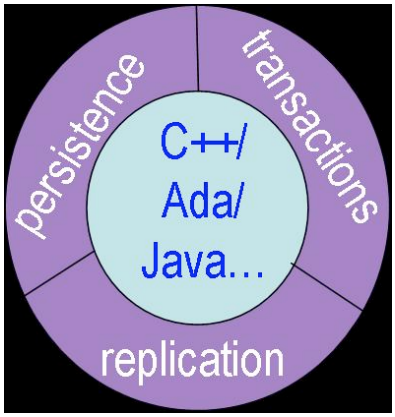


Figure 1: Object databases extend programming language capabilities

There are of course other ways to make objects persistent and enforce transaction semantics to ensure consistent updates to the objects, such as object/relational mappings. Object databases have a distinct advantage over these kinds of approaches, however, because they can directly use the type system of the object-oriented programming language. Eliminating the "mapping" process between the type system of the programming language and the type system of the database engine (e.g. SQL) offers significant performance advantages and can also reduce application development time. (Loomis 1995) defines the role of an object database this way:

"The fundamental role of an object DBMS is to provide persistent storage management for objects in a way that

1. allows for highly efficient but easy access from object programming languages,
2. hides the complexities of distribution of objects across network sites, and
3. allows multiple users and applications shared, protected access to the objects."

What makes an object-oriented database "real-time"?

The primary characteristic for true-real-time software is *determinism*. For a database engine, this implies that the worst-case execution times for a transaction must be predictable so that an application designer can guarantee that system timing constraints will be met. It is also desirable that the worst-case and best-case execution times for a transaction differ by as little as possible. Designing a database engine with these kinds of constraints in mind requires several significant architectural trade-offs, and this proscribes the possibility of creating a real-time object database by simply "tweaking" a non-real time one (Roark et al. 1996). Moreover, the kinds of systems that might use a real-time database engine are frequently embedded, and operation in an embedded environment levies additional requirements on the architecture of the database engine. The following are attributes that an object database must have in order to succeed in a demanding real-time, embedded environment:

1. Predictable, bounded transaction execution times- The concurrency control system, storage management, and data structures used by the database engine must enable the a priori computation of the worst-case execution time for a transaction using a specified

database schema containing a specified number of objects of a specified size. This allows a designer to structure the schema to ensure timelines will be met. A transaction here refers to a sequence of calls to the object database methods for accessing objects in the database, such as searching an index, iterating through a collection, creating or deleting database-managed objects etc.

2. Predictable, bounded resource consumption- The storage management and concurrency control systems used by the database engine must enable the a priori computation of the amount of storage required to hold a specified number of objects of a specified size. This allows a designer to determine the required storage "footprint" for the database engine.
3. Long-term unattended operation- The database engine must be able to operate 24/7 indefinitely without human intervention and without being taken off-line for storage defragmentation or other database maintenance.
4. Support for main-memory databases- The database engine must support the creation of databases with full transaction semantics, recovery, etc. in main memory as well as in secondary persistent storage (e.g. disks).
5. Efficient concurrency control- The concurrency control system used by the database engine must support efficient read/write access to objects and must minimize the possibility of priority inversion and mitigate it when it does occur (e.g. priority inheritance etc.).
6. Fault tolerance- The database engine must support, at a minimum, a "hot spare" redundancy policy to support uninterrupted operation in the event of a failure on the primary database host.

There are other attributes one could add to a list like this, but these are the fundamental ones required for successful database deployment in most real-time embedded systems.

The following sections describe the kinds of applications that use (or could use) real-time object database engines.

Real-time pattern recognition

Real-time pattern recognition is increasingly important as more and more information is digitized. Applications that make use of real-time database technology as part of a pattern recognition system cross the spectrum from real-time signature verification (BirdStep 2004) to electronic warfare (EW) (McDaniel and Schaefer 2003). The author has some experience in the EW domain, and it is among the most challenging in terms of performance requirements so that will be used to provide some context for the discussion of this topic.

EW systems are classical pattern recognition machines as depicted in the block diagram shown in Figure 2. This block diagram shows components from a surface ship EW system, which would typically be used for detecting and classifying threats like enemy aircraft and anti-ship missiles. The system uses sensors (antennae) to detect radar signals. The sensor outputs go to a pulse processor that typically includes a wideband RF receiver and a digitizer. The pulse processor is the feature extractor for the system: that is, it measures the characteristics of the received signals that will allow those signals to later be classified. Common measurements include the frequency,

amplitude, pulse duration, spacing between pulses, and modulation within and between pulses for these basic measures (Adamy 2001).

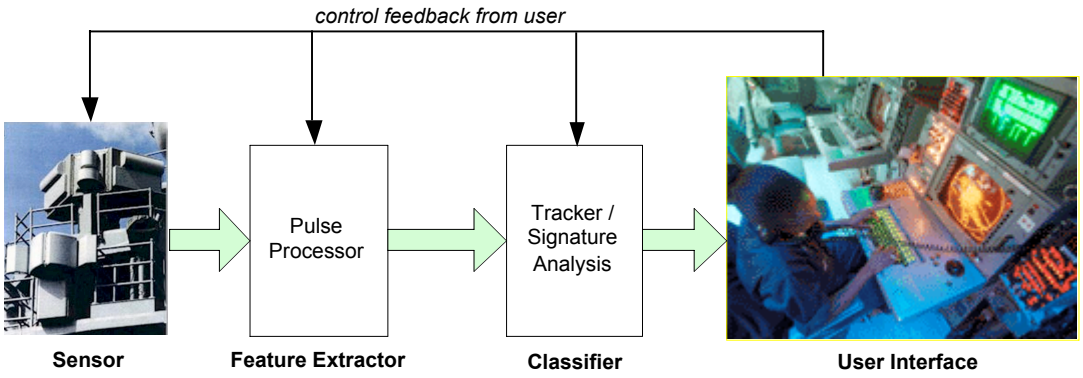


Figure 2: EW System Block Diagram

When the features of the intercepted emitter mode have been extracted by the pulse processor, the resulting feature vector is sent to the tracker/signature analysis function, which serves as the classifier for the system. It is here that the real-time search of the database of emitter signatures takes place. Finally, the resulting track and its threat classification are sent to the user interface for display, and typically a human operator can then adjust the operation of the system using available controls. This provides a feedback loop to optimize system performance.

Emitter modes typically are not a point in feature space, but rather occupy a volume. This is because modulation is commonly used on one or more of the key attributes of the mode. In parameter space, the mode can therefore be represented as a hypercube per Figure 3 below.

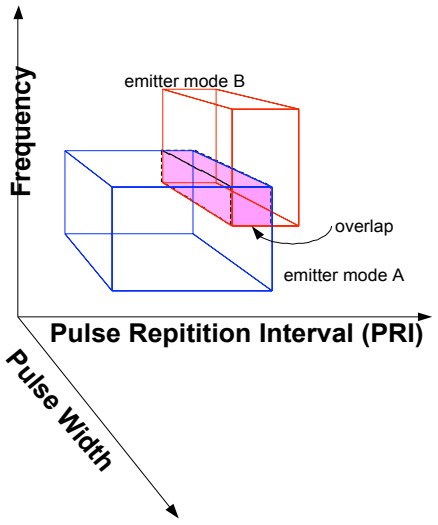


Figure 3: Emitter parameter hypercubes

Note that this means the emitter mode signatures are spatial data, which in turn implies that they do not have a partial ordering (i.e. you cannot define a $\leq$ operator for them) and therefore cannot be efficiently organized with a standard point access method (PAM) like a B-tree.

In the EW test described in (McDaniel and Schaefer 2003), feature vectors were presented to the classifier at a rate of 5 per second. The signature database contained emitter mode signatures for 3,000 kinds of emitters, and these emitters were in turn associated with the platforms (ships, planes, missiles, etc.) that carried them. The results of their tests showed a commercial real-time relational database with all data in RAM was able to support classification and find all of the possible platforms for the matching emitter modes in an average time of about 200 milliseconds, with the actual time varying between about 2 milliseconds and 5+ seconds as shown in Figure 5-3 in (McDaniel and Schaefer 2003).

While this performance is certainly better than a disk-based relational database, a real-time object database should be faster and more deterministic than the results reported in (McDaniel and Schaefer 2003). Part of the reason for this is that the object database can include R-tree based spatial indexes which would greatly accelerate range query searches of the emitter mode signatures database. Also, navigating through a complex schema like that shown in Figure 4-3 of (McDaniel and Schaefer 2003) is faster in an object database because the objects contain links to all other objects that they are related to. This obviates the need for costly joins or secondary key lookups to find related objects, and of course there is no need to transform objects between the type system of the application (C++, Ada, Java, etc.) and the type system of the database (SQL tables). Eliminating this kind of overhead has significant performance advantages for these kinds of applications. It would be interesting to repeat the experiment in (McDaniel and Schaefer 2003) with a real-time object database and quantify the differences.

Real-time control systems

Another domain where a real-time object-oriented database could be useful is that of real-time control systems. Of particular interest are those systems containing a distributed network of sensors, such as advanced building control systems. There are currently many systems available that monitor and control the indoor environment (e.g. the Heating, Ventilation, and Air Conditioning (HVAC) system), but the use of embedded database technology would make advanced capabilities like crisis management, evacuation, guidance of first responders, and counter-terrorism response possible. Consider the diagram shown in Figure 4:

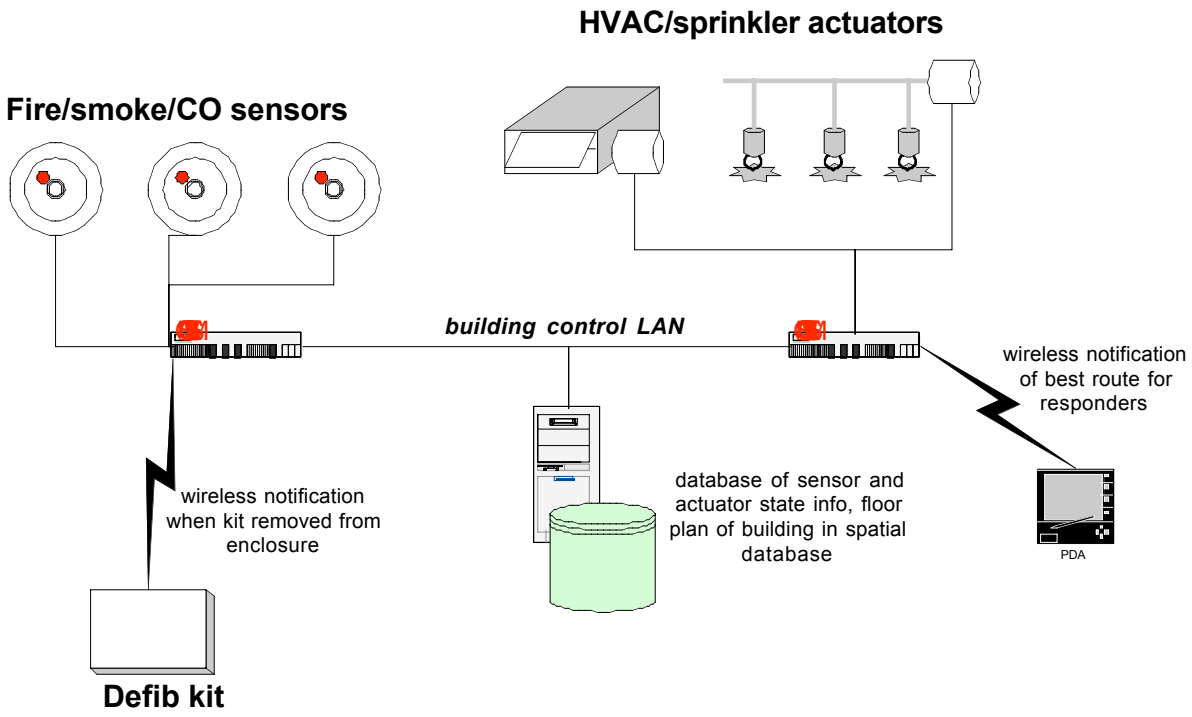


Figure 4: Database in building control system

Here, the building control system, based on an object database manager, communicates with a distributed network of sensors in the building via a building control "LAN," which could include both wired and wireless components. The object database could contain a "building schema" with the sensors, HVAC plenums, sprinkler heads, portable defibrillator kits, and other system components in a spatial index. This would allow the building control application to perform spatial query processing based on sensor events. For example, if some of the sensors report detection of smoke, the application can query the database for plenum ducts serving the area where the sensor is located and command the actuators to close or open the ducts as desired. This technique is useful not only for managing smoke or carbon monoxide from a fire, but it could also be used to counter other threats such as injection of a toxin into the building's HVAC system. Responses to threats like this are even more time-critical than responses to fires, and a real-time database is needed due to the large number of toxin sensors and the very low latencies required once detection occurs.

In addition to being able to operate the HVAC and sprinkler systems in response to sensor inputs, the embedded database could also provide best routes into the building for first responders. This could be done by storing the floor plans for the building into a spatial indexing structure and calculating the fastest and safest way to the trouble area being reported by the sensors. A corresponding set of instructions could be broadcast from the wireless network to handheld units, radios or other first responder equipment. This kind of capability is especially useful for handling the more routine forms of emergencies such as heart attacks or other health emergencies for building occupants. In Figure 4, the wall mount unit for a portable electronic defibrillator (PED) sounds an alarm and sends an alert to the network when its PED is removed.

Including PEDs, fire extinguishers or other safety devices with usage alarm capabilities in the database would enable the building control system to call the 911 center or direct the building's own emergency response team to the trouble area by the best route. The time saved in getting first responders to the scene could be the difference between life and death.

A complex building control system like the one in Figure 4 must maintain the state of the building. The embedded database would be responsible for updating the building "state vectors" over time. This requires fast processing of temporal data (e.g. sensor reading histories), and this has been a difficult problem when relational databases have been used in these kinds of systems (Olken et al. 1998). This is another area where object databases excel because their collection types are richer than the simple 2-D table used in relational products. An in-memory object database with a time-ordered circular queue collection would have addressed the performance issues described in (Olken et al. 1998).

A related problem in this domain is building security. Future building designs will likely incorporate biometric sensors to verify identity within the building. Each type of biometric sensor has its inherent weaknesses that can be mitigated using fusion techniques (Veeramachaneni et al. 2003). Figure 1 in (Veeramachaneni et al. 2003) shows a biometric security system based on a network of sensors. These sensors provide different kinds of biometric readings (e.g. face identification, fingerprints, voice identification, etc.) and supply an "accept/reject" decision to a Bayesian decision function. In order to supply an identity accepted/rejected hypothesis, however, the biometric sensors must have a database of feature vectors against which to compare their current input. This is where an embedded real-time object database could be used to construct the kind of system described in (Veeramachaneni et al. 2003). Figure 5 below shows the original Figure 1 from (Veeramachaneni et al. 2003) with the data management functions added.

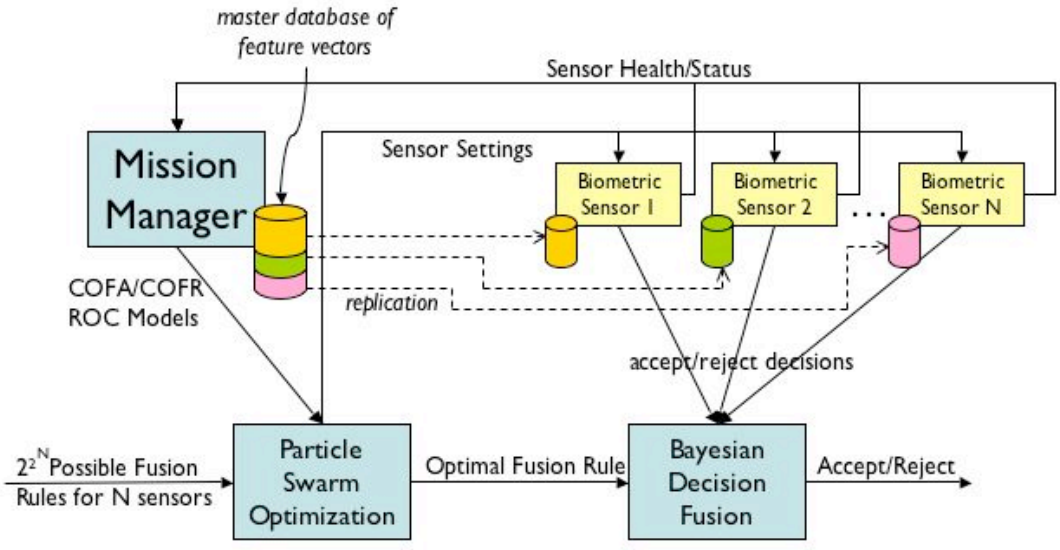


Figure 5: Biometric building security network with embedded ODBMS

In Figure 5, a master database of feature vectors is kept in the "Mission Manager" function. This database contains all of the feature vectors for all of the building's occupants. The database might contain sets of fingerprints, digital portraits, voice prints, or other forms of biometric identification information. Each biometric sensor could open a replicant of the relevant data and use this replicant to perform pattern-matching queries. The advantage of a configuration like this is that the sensors would then not have to query feature vector data from a central location (thus reducing network bandwidth). Also, this design allows for real-time updates to the master database so that, for example, building guests could be entered into the database and their feature vectors would be automatically propagated to the building's sensors. An embedded real-time object database is an ideal fit for such a system.

New real-time object database work: optimizing spatial indexing

The possible utility of real-time object databases for real-time pattern-matching applications has been described in previous sections. Use of an embedded database in these domains requires an efficient and robust dynamic spatial indexing mechanism. The data structure of choice for this kind of data has been the R-tree (Guttman 1984) and its variants, such as the R*-tree (Beckmann et al. 1990). Unfortunately, the R-tree and its variants do not perform well in "high dimensional" space because the bounding boxes for the tree's directory nodes begin to overlap significantly as the number of dimensions increases (Bohm et al. 2001). This causes so many nodes in the tree to be visited during a query that the performance becomes worse than a linear search. This problem does not generally occur for data with 3 or 4 dimensions, but multimedia data frequently have 10 or more dimensions. These dimensions represent color histograms and other statistical measures of images. The need to search databases of images for nearest matches in real time requires a spatial index with sublinear performance for a large number of dimensions.

The X-tree (Berchtold et al. 1996) was an early effort to mitigate the effects of "dimensional degradation" on the R-tree. The X-tree creates "super-nodes" in places in the R-tree where the directory nodes begin to overlap significantly. This essentially turns a portion of the tree into an array, which ensures that a query that accesses that part of the tree will not have performance that is worse than a linear search. The X-tree shares the one of the chief advantages of the R-tree, the ability to index both points and rectangles. Later efforts to retain this advantage and achieve sublinear performance (i.e. faster than a linear search) include the QSF tree (Orlandic and Yu 2000). The QSF tree maintains two "bounding boxes" for each index node. One bounding box (the "L" region) contains all of the "lower left" corners for all of the spatial keys in a node, and the other (the "H" region) contains all of the "upper right" corners for all of the spatial keys in a node. The "L" regions for the nodes are sorted using a kdB tree, and the tree is searched using the "lower left" corner of the query region. When a matching leaf node is found, the upper-right corner of the query region is checked against the "H" regions in matching leaf nodes to be sure it's a match. The QSF tree thus uses a PAM that has no overlap in its directory nodes, but which is capable of storing rectangles as well as points. This makes the QSF tree an excellent candidate for a spatial index in an object database.

Another option for complex, high-dimensionality data would be to use a Support Vector Machine with a Gaussian kernel to find clusters in the data (Frailis et al. 2003). The clusters

found by the SVM could then be inserted into a spatial index for high-dimensional data that is designed to sort clusters, such as the Cluster Tree (Yu and Zhang 2003). The SVM could be deployed as a database application and be optimized for use in a real-time system, where it would serve as a "cluster-finder." The resulting clusters could then be stored in a Cluster Tree. The clusters which are initially loaded into a Cluster Tree can be thought of as its "training set" and new data points can be directly inserted into the tree without the SVM figuring out what cluster they belong to. A system like this could be very useful for doing pattern recognition and analysis on well-known kinds of data where a training set of clusters could be loaded into the Cluster Tree. For more dynamic environments, the application would need to have a mechanism to determine when the Cluster Tree would need to be "re-trained."

Conclusion

Embedded real-time applications are performing increasingly more complex processing as available CPU power, network bandwidth, and storage speed increases while power consumption for these chipsets decreases. The need for robust database capability in these systems is clear. The programming languages currently available for the construction of these systems do not include transaction semantics or seamless persistence. Real-time object databases are therefore an excellent choice to meet the data management needs for these systems, as they minimize both the development time and resource utilization for the database application(s) in the system. Object databases also can accommodate new and novel indexing schemes for dealing with complex, "high dimensionality" data. This kind of data is becoming more common in databases as the need arises to do biometric identification and real-time image matching, and real-time object databases are a superior technology for the ideal accessing and managing this information.

References

Adamy, David (2001); EW101- A First Course in Electronic Warfare; Artech House radar library (pp. 73-103)

Beckmann, Norbert, Kriegek, Hans-Peter, Schneider, Ralf, and Seeger, Bernhard (1990); The R*-tree: An efficient and robust access method for points and rectangles; Proceedings of the ACM SIGMOD International Conference on the Management of Data (pp. 322-331)

Berchtold, Stefan, Keim, Daniel A., and Kriegel, Hans-Peter (1996); The X-tree: A Data Structure for High-Dimensional Data; Proceedings of the 22nd VLDB Conference (pp. 28-39)

BirdStep Corporation (2004); company's external web site http://birdstep.com/solutions/database_enduser.php3; see use of BirdStep real-time object database in AutoSig Systems, Inc. automatic signature verification

Bohm, Christian, Berchtold, Stefan, and Keim, Daniel A. (2001); Searching in High-Dimenssional Spaces: Index Structures for Improving the Performance of Multimedia Databases; ACM Computing Surveys, Vol. 33, No.3 (pp. 322-373)

Chakrabarti, Kaushik, and Mehrotra, Sharad (2000); The Hybrid Tree: An Index Structure For High-Dimensional Spaces; Proceedings of the 15th International Conference on Data Engineering (pp. 440-447)

Frailis, Marco, De Angelis, Alessandro, and Roberto, Vito (2003); Data Management and Mining in Astrophysical Databases; A Non-linear World: the Real World- Second International Conference on Frontier Science (pp. 1-22)

Gamma, Erich, Helm, Richard, Johnson, Ralph, and Vlissides, John (1994); Design Patterns: Elements of Reusable Object-Oriented Software; Addison-Wesley (pp. 257-272)

Guttman, Antonin (1984); R-Trees: A Dynamic Index Structure for Spatial Searching; Proceedings of the ACM SIGMOD International Conference on the Management of Data (pp. 47-57)

Loomis, Mary E.S. (1995); Object Databases: The Essentials; Addison-Wesley (pp. 15)

McDaniel, Dave and Schaefer, Gregory (2003); Real-time DBMS for Data Fusion; Proceedings of the 6th International Conference on Information Fusion (pp. 1334-1341)

Olken, Frank, Jacobsen, Hans-Arno, McParland, Chuck, Piette, Mary Ann, and Anderson, Mary F. (1998); Object Lessons Learned from a Distributed System for Remote Building Monitoring and Operation; Proceedings of OOPSLA 1998 (pp. 288)

Orlandic, Ratko, and Yu, Byunggu (2000); A Study of MBR-Based Spatial Access Methods: How Well They Perform In High-Dimensional Spaces; Proceedings of the 2000 International Database Engineering and Applications Symposium (IDEAS ’00) (pp. 306-315)

Roark, Mayford B., Bohler, Michael and Eldridge, Barbara L. (1996); Embedded Real-Time and Database: How Do They Fit Together?, Proceedings of the Software Technology Conference 1996, Track 9, Embedded Systems (pp. 6-13 of this paper)

Veeramachaneni, Kaylan, Osadciw, Lisa, and Varshney, Pramod K. (2003); Multisensor Surveillance Systems: The Fusion Perspective; Kluwer Academic Publishers (pp. 265-286)

Yu, Dantong and Zhang, Aidong (2003); ClusterTree: Integration of Cluster Representation and Nearest-Neighbor Search for Large Data Sets with High Dimensions; IEEE Transactions on Knowledge and Data Engineering, vol. 15, no.5 (pp. 1316-1337)

Model-based Data Management for Mediation Services for Intelligent Software Agents

Andreas Tolk, Ph.D.
Virginia Modeling Analysis and Simulation Center
Old Dominion University
Norfolk, VA 23529, United States
+1 (757) 686 – 6203
atolk@odu.edu

Johnny J. Garcia
General Dynamics/ AIS
7025 Harbour View #101
Suffolk, VA 23435, United States
+1 (757) 673 – 2411
john.garcia@gd-ais.com

Keywords

Service-oriented Architectures, Extensible Markup Language (XML), Command & Control Information Exchange Data Model (C2IEDM), Mediation Services, Global Information Grid (GIG)

Abstract

Mediation services can be generally defined as a mechanism to map interchange formats (map them to what?), thus increasing the ability for disparate systems to exchange information through common methods. However, when intelligent software agents use these meditation services, syntactical translations of formats are not sufficient.

The semantic context has to be captured and interchanged as well; a common ontology is needed as the basis for the mediation service. While in the commercial world several recent publications are looking at possible automated solutions, in complex environments (is the commercial world not complex?), data engineering is necessary in order to support semantically meaningful mediation layers. Model-based data management uses a common reference model to map data models to data sources to support intelligent software agents for their internal decision processes

This paper defines the phases of data engineering, shows potential conflicts and how they can be solved, and gives an example from the military application domain by showing how the Command and Control Information Exchange Data Model (C2IEDM) developed by the North Atlantic Treaty Organization (NATO) can be used as a common reference model for military applications.

Overview

In order to support operations with rapidly changing requirements, service oriented architectures are being developed in lieu of the often too inflexible traditional solutions. As an alternative to having a system fulfilling a set of predefined requirements, services fulfilling requirements are identified, composed, and orchestrated to achieve the current users' needs in an ongoing operation. Grid Computing, System-of-Systems engineering approaches, and the Global Information Grid are examples of this trend. Intelligent agents are required to identify the service requirements in a given situation, find applicable services, compose these services to support the operation, orchestrate their execution, and evaluate the result in a way meaningful to the user.

One of the most urgent requirements in this circle is to ensure meaningful interoperability of the information exchanged between the services. This is a real challenge during the design and im-

plementation of the service since the designer does not know with which other services this service will communicate. This can only be determined during runtime. Although in the commercial world, some applications try to do such mappings automatically, their domains are relatively simple, such as address mapping problems (Su et al. 2001). For more complex problem sets, explicit mapping of data must be done. A common way must be used to capture the format (syntax), meaning (semantics), and the use (pragmatics) of data in order to avoid ambiguity and structural variances when composing the services. This is the domain of model based data management. The results can be used to establish the required mediation layers necessary for the intelligent agents to support the process effectively.

Service Oriented Architectures

Traditional information technology (IT) followed a waterfall model starting with a set of user requirements, which led thru several stages to the system definition, system design, and system implementation. The reality of required distributed computing and the necessity for combining information resources using very heterogeneous IT infrastructures – different hardware, middleware, languages, etc. – cannot be met by such traditional efforts. Starting with the ideas of net-centric operations and setting up a system of systems, the commercial world, as well as the military world, is moving from system components delivering the operationally required functionality, towards service oriented architectures. Within the commercial world, distributed computing environments operate as a uniform service, which looks after resource management and security independently of individual technology choices. Grid computing is a means of network computing that harnesses the unused processing cycles of numerous computers to solve intensive problems that are often too large for a single computer to handle. In other words, grid computing enables the virtualization of distributed computing and data resources such as processing, network bandwidth, and storage capacity, to create a single system image which grants users and applications seamless access to vast IT capabilities. Just as an Internet user views a unified instance of content via the Web, a grid user essentially sees a single, large virtual computer. In order to access the functionality, services are defined based on common open standards and bridge the gap between the heterogeneous worlds of different languages, middleware solutions, and hardware. The authors perceive web services to be currently the strongest candidate for a technical solution to instantiate a service-oriented architecture.

This trend can be observed in the military world. Following the ideas of net-centric warfare (Alberts and Hayes, 2003), future military operations will be characterized by the seamless sharing of information and other resources. The technical backbone enabling this vision will be the Global Information Grid (DoD, 2002), which will be implemented using the Internet Protocol version 6 as the technical baseline. It will establish a service-oriented architecture of military services, from command and control to modeling and simulation, supporting the soldiers in all relevant military operations.

The real potential of service oriented architectures lies in the possibility to compose services and to orchestrate their execution, thus enabling new functionality compositions to fulfill the current often changing user requests within an ongoing operation. Information must be exchangeable between all composed services in a meaningful manner, i.e., not simply exchanging bits and bytes, but ensuring the interpretation of data in a consistent way leading to the same information, knowledge, and ultimately awareness within the services and their users; each service has to

know what data is located where, the meaning of data and its context, and into what format the data has to be transformed to be used in respective services composed into a distributed application within the overall system. To generate the answers to these questions is the objective of data administration, data management, data alignment, and data transformation. These can be defined as the building blocks of a new role in the interoperability process: Data Engineering (Tolk, 2004). The composing terms are defined as follows:

Data Administration is the process of managing the information exchange needs that exist between the services, including the documentation of the source, the format, context of validity, and fidelity and credibility of the data. Data Administration therefore is part of the overall information management process for the service architecture. Data Management is planning, organizing and managing of data by defining and using rules, methods, tools and respective resources to identify, clarify, define and standardize the meaning of data as of their relations. Data Alignment ensures that the data to be exchanged exist in the participating systems as an information entity or that the necessary information can be derived from the data available, e.g., using the means of aggregation or disaggregation. Finally, Data Transformation is the technical process of aggregation and/or disaggregation of the information entities of the embedding systems to match the information exchange requirements including the adjustment of the data formats as needed.

Model based data management uses a reference model to capture the planning, organizing and managing of data. Instead of mapping point-to-point, the information exchange requirements of a service are mapped to a common information exchange reference model, which can be seen as the ontology of the application domain.

C2IEDM

In 1978, NATO's Long-Term Defense Plan (LTDP) Task Force on C2 recommended that an analysis be undertaken to determine if the future tactical Automatic Data Processing (ADP) requirements of the Nations (including that of interoperability) could be obtained at a significantly reduced cost when compared with previous approaches. In early 1980, the then Deputy Supreme Allied Commander Europe initiated a study to investigate the possibilities of implementing the Task Force's recommendations. This resulted in the establishment of the Army Tactical Command and Control Information System (ATCCIS) Permanent Working Group (APWG) to deal with the challenge of the future C4I systems of NATO. The ATCCIS approach was designed to be an overall concept for the future command and control systems of the participating nations. One constraint was that each nation could still build independent systems. To meet this requirement, ATCCIS defined a common kernel to facilitate common understanding of shared information. In 1999, ATCCIS became a NATO standard with the new name Land Command and Control Information Exchange Data Model (LC2IEDM). In parallel to this, the project managers of the Army Command and Control Information Systems (C2IS) of Canada, France, Germany, Italy, the United Kingdom, and the United States of America established the Multilateral Interoperability Program (MIP) in April 1998. By 2002, the activities of LC2IEDM and MIP were very close, expertise was shared, and specifications and technology were almost common. The merger of ATCCIS and MIP was a natural and positive step. LC2IEDM became the data model of MIP. Finally, in 2003 the name was changed to Command and Control Information Exchange Data Model (C2IEDM).

There are two application domains for the C2IEDM within NATO: Data Management and Information Exchange. The NATO Data Administration Group used the C2IEDM to map all information exchange requirements between the national command and control systems to it in order to ensure semantic (What do the data mean?) and pragmatic (What are the data used for?) interoperability between the systems. The MIP data managers will continue this task after the merger between MIP and C2IEDM is finished. MIP also uses the C2IEDM to exchange data between national command and control systems in order to foster sharing information and gain a common understanding on what is happening on the battlefield. To this end, the national systems establish data translation layers mapping their internal data presentation to the data elements of C2IEDM for information exchange with the other systems.
A technical view on the data model goes far beyond the scope of this paper, as C2IEDM comprises data elements describing a common vocabulary consisting of 176 information categories that include over 1500 content elements. In order to cope with these needs, C2IEDM is divided into a Generic Hub comprising the core of the data identified for exchange across multiple functional areas. It lays down a common approach to describe the information to be exchanged and is not limited to a special level of command, force category, etc. In general, C2IEDM describes all objects of interest on the battlefield, e.g., organizations, persons, equipment, facilities, geographic features, weather phenomena, and military control measures such as boundaries. In addition to this, special functional areas are defined extending the Generic Hub under national responsibility to cope with information exchange needs of national concern. A tutorial on C2IEDM is given in (Loaiza, 2004). The complete data model documentation and additional information is available on the Internet (MIP, 2004).

C2IEDM for Intelligent Agents

The use of C2IEDM for intelligent agents to collaborate and support within the Global Information Grid is already presented in (Pohl, 2004). The authors share this view and are motivating this as follows: The application of the Extensible Mark-up Language (XML) enabled a new level of interoperability for heterogeneous IT systems. However, although XML enables separation of data definition and data content, it doesn't ensure that data exchanged is interpreted correctly by the receiving system. A common reference model defining the XML tag sets and the structure is needed to ensure meaningful interoperability. For military operations, the C2IEDM has the potential to become such a reference model. Intelligent agents can use both approaches. C2IEDM structured in XML schemas can become the "language" spoken by the agents and used to communicate between agents and services. As the operational use is also part of the C2IEDM agreements, even pragmatic interoperability can be reached; the C2IEDM becomes an ontological layer for the GIG. The ultimate use is that the combination of services, agents, and the common ontology enables a quantum leap in command and control quality, as described in (Alberts and Hayes, 2003): When data are put into context, the result is information; applied information in form of procedural access leads to knowledge; and knowledge applied to analyses leads finally to awareness. When the text based messages where replaced by a common operating picture, we stepped up from data to information. The introduction of modeling and simulation services introduces procedural knowledge. If intelligent software agents can now use these services and map them to observations of the real world, they can support the analyses of military experts.

Hence, the command of control systems incorporating agents, services, and a common ontology can even support awareness, which traditionally is seen in the cognitive domain only.

Summary

Service-oriented architectures are evolving rapidly. In order to make better use of the services delivering the necessary functionality, intelligent agents are required. The agents need to make sense of the services functionality, their use of data, and their behavior. To this end, effective (metadata has not been mentioned up to the point, why is it mentioned in the summary???) metadata and meta-model management is necessary. One of the most challenging tasks in this context is the management of information exchange requirements of services in a way that these services can be composed and orchestrated with other services during runtime delivering the functionality as specified by user during the ongoing operation. In the military environment, the Global Information Grid is the technical backbone. The Command and Control Information Exchange Data Model (C2IEDM) is a matured approach for a military ontology in the domain of command and control. C2IEDM has been proven to be flexible enough to cope with all information exchange requirements of services. Technically, the definition of mediation layers to make this information available to intelligent agents is feasible. XML can be used as a syntax layer, the C2IEDM definitions can be used for ensuring semantic interoperability, and the C2IEDM structures and views – which have been agreed to by military operators of the developing nations – can insure that the pragmatic view, i.e. how the data is used – is aligned as well.

References

Alberts, D. and Hayes, R. (2203); Power to the Edge – Command, Control, in the Information Age; Command and Control Research Program Publications

Department of Defense (2002); DoD Directive 8100.1: Global Information Grid (GIG) Overarching Policy; The Pentagon, Washington, D.C.

Loaiza, F. (2004); C2IEDM Tutorial; Proceedings of the 2004 C2IEDM Workshop, Fairfax, VA

MIP (2004); Multilateral Interoperability Programme website; http://www.mip-site.org [last visited June 2004]

Parent, C. and Spaccapietra, S. (1998); Issues and approaches of database integration; Communications of the ACM; Volume 41 (5), pp. 166 - 178

Pohl, J. (2004); C2IEDM as a Component of the GIG Architecture; Proceedings of the 2004 C2IEDM Workshop, Fairfax, VA

Spaccapietra, S., Parent, C. and Dupont, Y. (1992); Model Independent Assertions for Integration of Heterogeneous Schemas; Very Large Database (VLDB) Journal, Vol. 1 (1), pp. 81-126

Su, H., Kuno, H. and Rundensteiner, E. (2001); Automating the transformation of XML documents; Proceedings of the third international workshop on Web information and data management; Session: Web Information Integration, pp. 68-75

Tolk, A. (2004); Common Data Administration, Data Management, and Data Alignment as a Necessary Requirement for Coupling C4ISR Systems and M&S Systems; Journal on Information & Security, Vol. 12

Drill-Down Operator in Information Systems

C. Putonti, B. Czejdo
Loyola University
New Orleans, La. 70118, USA
czejdo@loyno.edu

M. Baszun
Warsaw Technical University, Warsaw, Poland
e-mail: baszun@imio.pw.edu.pl

J. Czejdo
Edinboro University of Pennsylvania
Edinboro, Pa. 16412, USA
jczejdo@edinboro.edu

Abstract

The decision-making process very often requires the ability to "drill down data" across different subject areas and dimensions. Data warehouses typically include this feature; however the implemented drill-down operators do not have enough flexibility. In this paper, we consider the problem of generalization of the drill-down operator for an information system. We introduce the concept of a general aggregation graph that is a good basis for defining all options for the drill-down operator. The general aggregation graph can be reduced to obtain the personalization graph that is specific for a user. This personalization graph allows for a customized drill-down operator.

Keywords

information system, drill-down operator, aggregation, data warehouses, personalization

1. Introduction

The proper infostructure is needed for rapid and accurate decision-making process. This process can be significantly improved when an information system allows for a convenient traversal of multi-level data by a human user or a software agent. In a limited way this function is accomplished in data warehouses (Bischoff et al.1997, Gupta et al. 1995, and Widom 1995) by a drill-down operator. Typically this feature falls short of its potential. Drill-down operators usually restrict the user, limiting the possibilities of different views. The effective use of such an operator can be improved if all options are available to the user and information about these options is clearly presented to the user

(Czejdo 2000). Additionally, personal options need to be automatically or manually specified as a priority list.

In this paper, we discuss the theoretical aspects and implementation of a personalized drill-down operator. We first introduce the concept of a materialization graph. The materialization graph includes fact table and the set of summary tables included to improve the performance of key applications [Czejdo 2000]. The fact table and all of its possible aggregated views are called the general aggregation graph. The materialization graph is a subset of the general aggregation graph. The general aggregation graph can define main options for the drill-down operator. By selecting the options relevant for a specific user or a group of users, a personalization graph can be created. This graph also addresses security problems by restricting some users from selected options.

In addition we discuss how the drill-down operator can be extended and applied to other information systems. We describe the implications of subclass hierarchy and recursive relationships for the general aggregation graph.

This paper is organized as follows. In Section 2, we describe a star schema on the example of the TurboMachine Company, defining various summary tables that could be included in the data warehouse. In Section 3, we extend a star schema to include subclasses and recursive relationships. Section 4 presents sample queries for the general aggregation graph and the development of the drill-down operator. Section 5 provides examples of the TurboMachine Company illustrating a portion of a flexible drill-down operator. In Section 6, the process of developing a personalization graph and refining a drill-down operator is discussed.

2. Aggregation graph for a star schema

Let us consider a data warehouse for the TurboMachine Company that has several branches with separate databases. Each branch lends machines to other companies (other companies referred to here as locations) and collects (usually once a week) the leasing fee based on the meter describing the extent of use of each machine.

The data warehouse schema for the TurboMachine Company is a relatively typical star schema as shown in Figure 1. The model includes dimension tables: Machine, Location, and Time, and a fact table, T11. The role of each table is as follows. **Machine** contains and keeps records of the information for a specific machine. **Location** contains the information about each company which has/had leased a machine. Here, we assume that **Time** is only a conceptual table since all time attributes can be identified in the fact table T11.

T11 has a log of all collections made at a location for a date and for a specific machine. T11 has five key attributes LNumber, Year, Month, Day, MID referred to also as index

attributes. The additional attributes are called aggregate attributes. In our example, for simplicity, we have only one aggregate attribute Money.

Very often, the performance of a data warehouse with only one fact table is not acceptable. Summary tables need to be introduced in addition to the main fact table T11 in order to improve performance. Therefore, it is important to develop and follow some rules in the design of a data warehouse with frequent aggregate queries. Let us first look at an example of summary tables T23, T24, T35, T36, and T43 as shown in Figure 2.

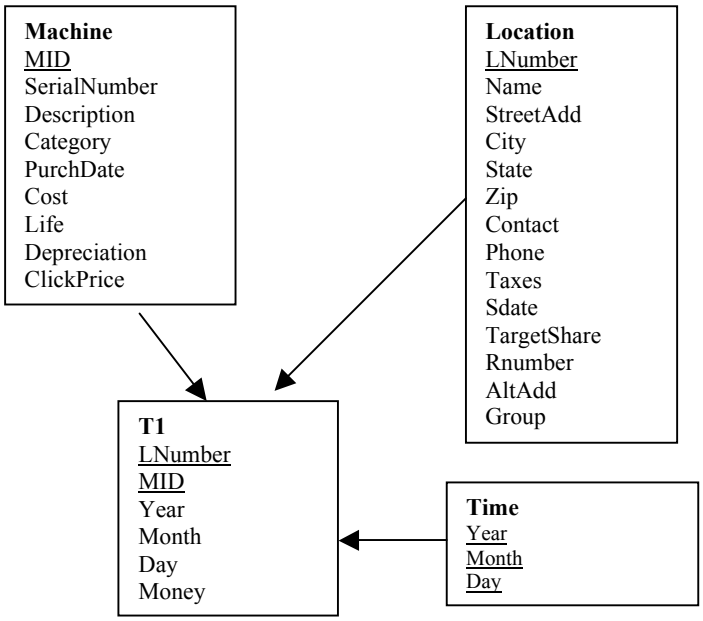


Fig. 1. The initial star schema for the TurboMachine data warehouse.

The additional summary tables are obtained by grouping T11 by its various index attributes. Table T23 is obtained by grouping T11 by index attributes LNumber, Year, Month and Day, and aggregating Money to obtain SumOfMoney.

```
CREATE TABLE T23 (LNumber, Year, Month, Day, SumOfMoney) AS
SELECT LNumber, Year, Month, Day, SUM(Money)
FROM T11
GROUP BY LNumber, Year, Month, Day;
```

Similarly we can create table T24 by grouping T11 by index attributes LNumber, MID, Year and Month, and aggregating Money to obtain SumOfMoney.

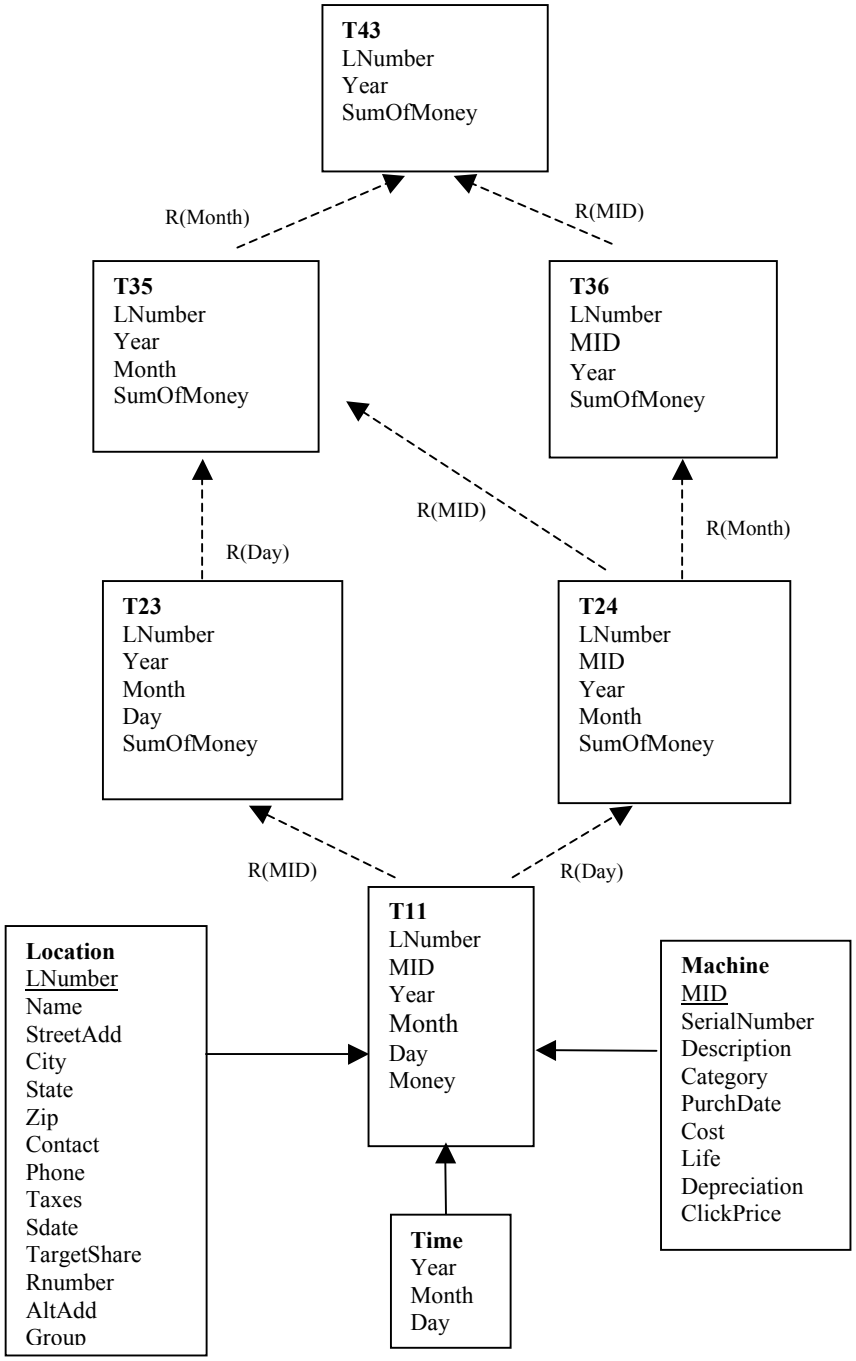


Fig. 2. Materialization graph for the TurboMachine data warehouse.

```
CREATE TABLE T24 (LNumber, Year, Month, MID, SumOfMoney) AS
SELECT LNumber, MID, Year, Month, SUM(Money),
FROM T11
GROUP BY LNumber, MID, Year, Month
```

The next level (third level) tables can be created from table T11 from the first level or from some already computed tables from the second level if the aggregate operator is of the type SUM, COUNT, etc. For example, to create table T35 we can use table T11, T23, or T24. If we choose table T24 then we need to group T24 by the index attributes LNumber, Year and Month, and aggregating Money to obtain SumOfMoney.

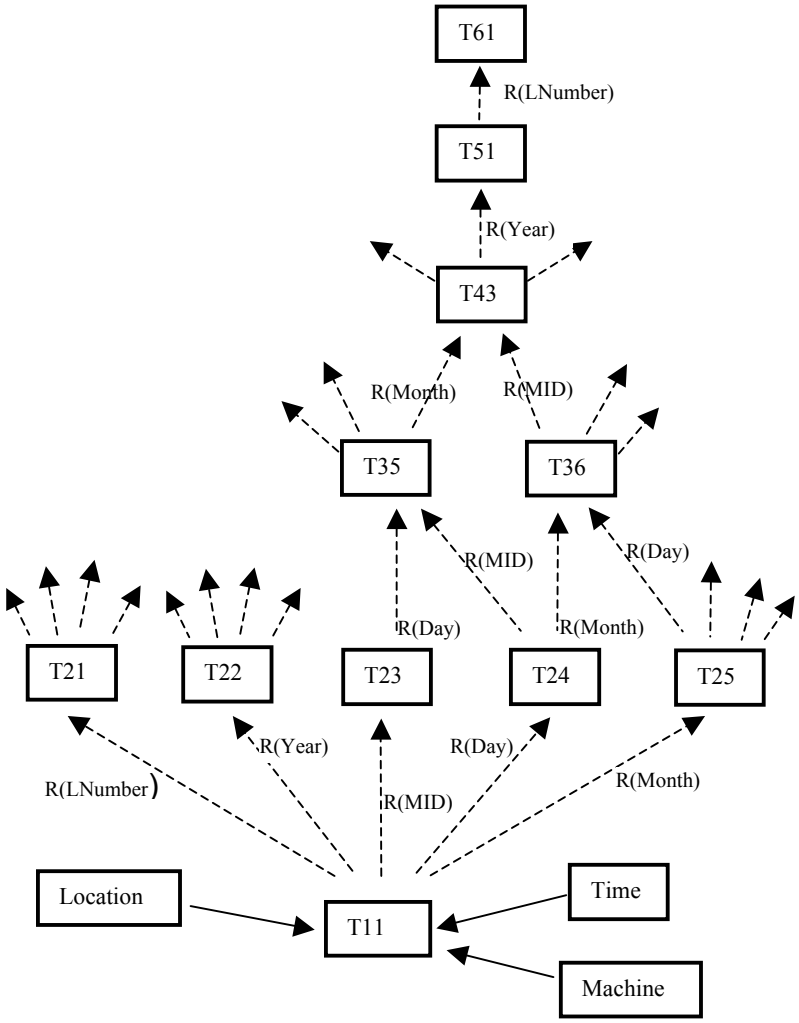


Fig. 3. General aggregation graph of the TurboMachine data warehouse

```
CREATE TABLE T35 (LNumber, Year, Month, SumOfMoney) AS
SELECT LNumber, Year, Month, SUM(SumOfMoney),
FROM T24
GROUP BY LNumber, Year, Month
```

In a similar manner, we can create tables on all higher levels. This model offers four levels of granularity. At a level 1 (lowest level), the granularity is the finest; there are many records due to the fact that each is unique by a specific date. When these records are summarized, the level of granularity is coarser. For example T24, a monthly summary, is at a higher level then T11 since its granularity is coarser. Furthermore, T36 and T43 are even higher. In general, coarser levels of granularity do not require as many records to be stored. Transformation of a table from a lower level to a higher level is accomplished by grouping by all but one index attribute. Since during this operation the singled out index attribute is removed, we will refer to this operation as R with the appropriate argument, what is also shown in Figure 2.

The schema of the TurboMachine Company uses the SUM aggregate to add the amount of money collected. Other aggregates, such as MIN, MAX, AVG, etc, should also be considered. The mathematical definition of MIN, MAX, and AVG are very different from SUM. A summation is possible on quantities that can either be one single instance or a sum of instances. It is not necessary to differentiate between the two. Minimums, maximums, and averages require the computation to be made for single instances.

For example, if the view of the year and the average amount of money is drilled down to the year, location, and the average amount of money, the derivation of this quantity requires the base table, T11. An average requires the amounts collected as well as a count of each collection. This is recorded in T11. In order to create each table, the query generated must be from T11.

```
CREATE TABLE T43(LNumber, Year, AvgOfMoney) AS
SELECT LNumber, Year, AVG(Money)
FROM T11
GROUP BY LNumber, Year
```

When aggregates other than SUM are used, a differing approach is necessary to create tables and generate views. Instead of aggregating from the highest possible table, the aggregation must be made from the base table, T11. Attributes of dimensions such as Location and Machine can be also used for aggregations resulting in the new nodes will be added to the aggregation graph. For example, in the Machine table the attribute Category can be used for aggregation. Consider that Category has two values, i.e., Video and Skill. Video refers to electronic video games and Skill refers to games such as pool, foosball, or air hockey.

3. Aggregation graph for a schema with subclasses and recursive relationships

By including aggregations by dimension attributes the number of nodes in aggregate graph increases. Figure 4 is the fragment of Figure 3, which takes into account the possibility of aggregation by the dimension attribute Category.

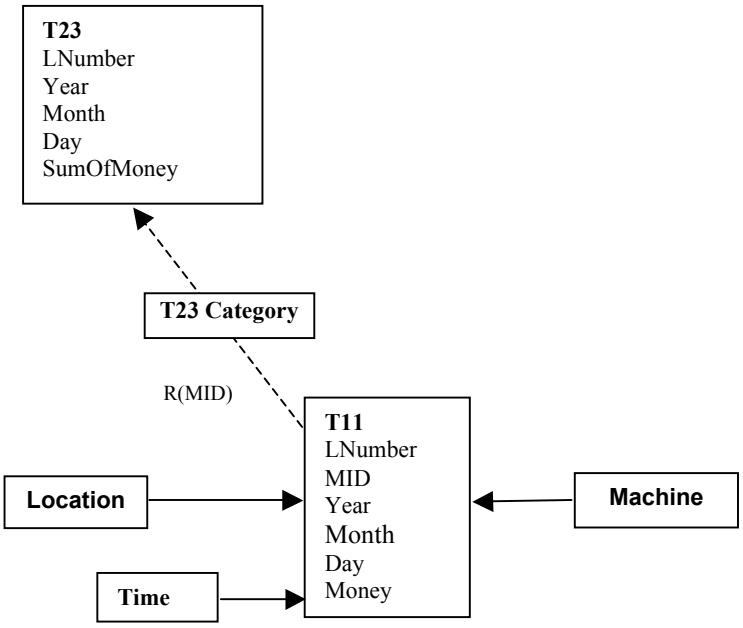


Fig. 4a. Fragment of a graph with aggregates based on subclasses.

The same approach can be used when we introduce the subclass concept. For example, Location and Machine have been treated as one class in the creation of the TurboMachine Company data warehouse schema, Figure 2. However, either one could consist of subclasses. The Machine table can divided into two separate classes, i.e., Video and Skill.

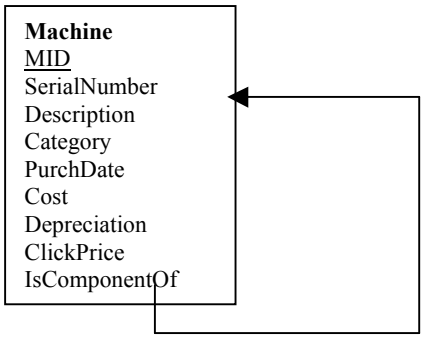


Fig. 4b. Fragment of a graph with a recursive relationship.

We encounter slightly different problems when a recursive relationship is present. For example, let us assume that each machine can contain other machines as is shown in Fig. 4b. The new attribute IsComponentOf is a foreign key defining the recursive relationship. In such situation we can create an explicit hierarchy of components such as level 1, 2, etc and build the aggregation graph based on such hierarchy.

4. Queries and Drill-Down feature

Each query can be associated with the specific node of a general aggregation graph. For example, let us consider a sample query Q35: "Find names for locations and the sum of money they made in December 1999".

```
SELECT Location.Name, SumOfMoney
FROM T35, Location
WHERE Location.LNumber=T35.LNumber
AND Year = 1999
AND Month = 'December';
```

This query can be associated with the summary table T35 and therefore was called Q35. Similarly, sample query Q36: "Find machine categories for all machines that made more than $5000 in 1999 while being in location with location number 234", shown below is associated with table T36.

```
SELECT Machine.Category
FROM T36, Machine
WHERE T36.MID = Machine.MID
AND Year = 1999
AND LNumber = 234
AND SumOfMoney > 5000;
```

Also sample query Q51: "Find names of locations that made in their lifetime more than $1000000", shown below

```
SELECT Location.Name
FROM T51, Location
WHERE Location.LNumber=T51.LNumber
AND SumOfMoney > 1000000;
```

is associated with table T51. If the tables T35, T36, and T51 were available, then, obviously, the execution of the above queries would be much quicker. If these tables were not available then the query processing would require grouping. Generally, we assume that either the table exists or that the appropriate view is defined. For example let

us assume that table T51 is not present and that the view is defined based on available tables in Figure 2, as shown below

```
DEFINE VIEW T51(LNumber, SumOfMoney) AS
SELECT LNumber, SUM(SumOfMoney)
FROM T43
GROUP BY Year;
```

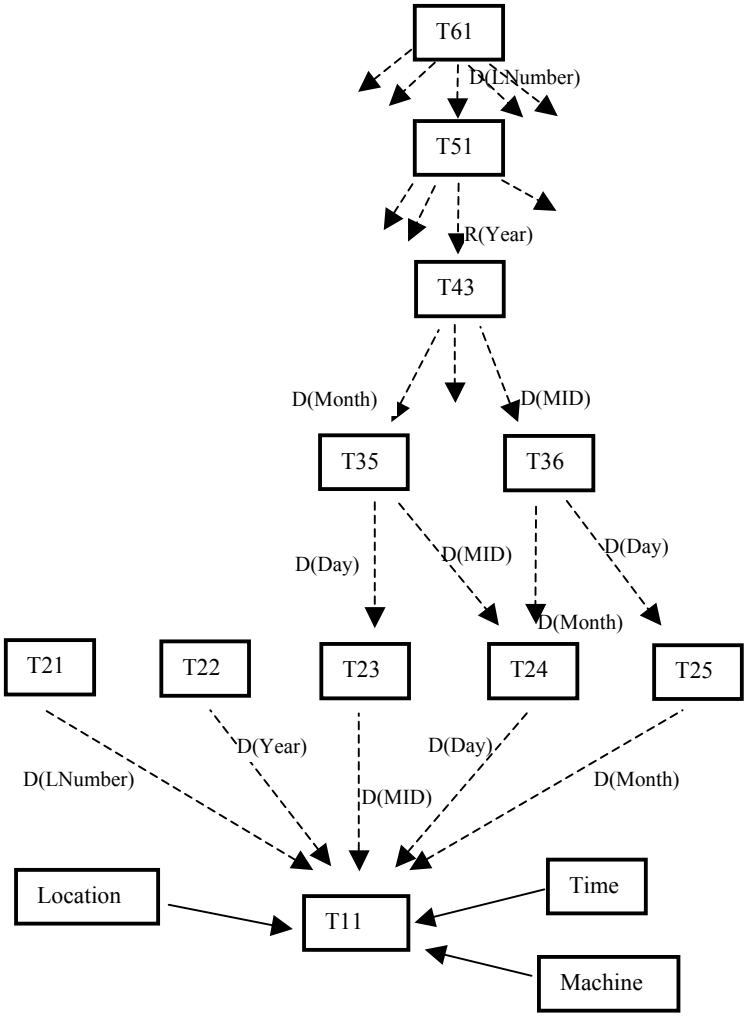


Fig. 5. General drill-down graph for TurboMachine data warehouse.

The general aggregation graph is crucial to the development of the drill-down operator. It can be easily converted to drill-down graph as shown in Figure 5. Each drill-down

operation is denoted by D(A) and has a single argument, A in general. The drill-down operation is the opposite of the aggregation that removes an index attribute. Here, instead, the table is expanded and the attribute A is added to the higher level table. Figure 5 shows the general drill-down graph. This graph depicts all possible groups of views corresponding to all index attributes not present at this level.

5. Implementation of drill-down operation

The use of the flexible drill-down operator can be illustrated by using the Microsoft Access database created for the TurboMachine data warehouse. The general aggregation graph defines multiple possibilities for the drill-down operator, but here only two possibilities available from T43 will be considered. These two possibilities correspond to the fragment of the drill-down graph shown in Figure 6.

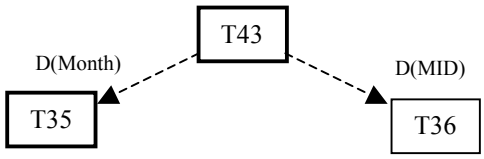


Fig. 6. Fragment of a drill-down graph.

Two different methods of drilling down will be considered. First, drilling down will be accomplished using a one record selection. Multiple record selections will be discussed later.

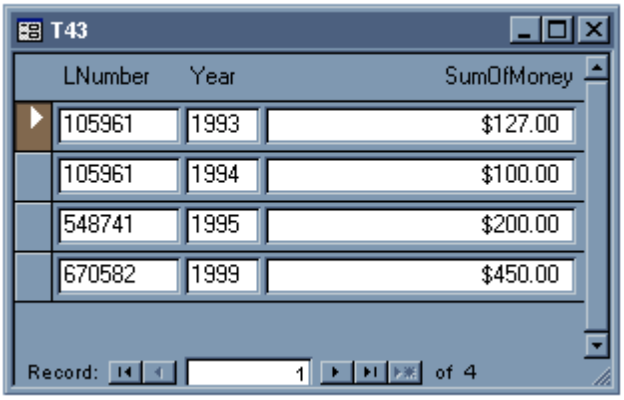


Fig. 7. T43 view.

The T43 table would be accessible through a form. Figure 7 is a sample form of the T43 table as well as the query selecting the view. To execute the drill-down operator in our

Access application, first, the argument for the drill-down operator needs to be specified from the given list.

More specifically, when selecting one record, such as the first record LNumber 105961 for 1993, a menu box with the options would appear. Those options include viewing monthly summaries for this location and year (T35) or viewing yearly summaries for all machines at this location for this year (T36). The user has flexibility in choosing the path he or she desires.

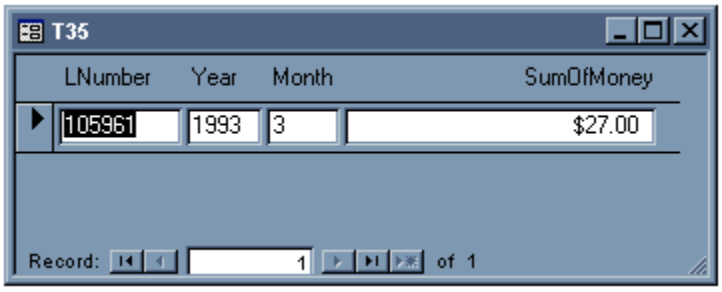


Fig. 8. T35 view.

If the user were to choose to view the monthly summaries for the LNumber 105961 in 1993, that record would be selected and the drill-down operator with the option for the attribute Month (T35) would be chosen, resulting in the view shown in Figure 7. Here for simplicity we assumed that data is available only for March and therefore only one record is displayed.

The view shown in Fig. 8 is obtained by constructing the SQL query with the WHERE clause "*LNumber = 105961* AND *Year = 1993"* as follows.

SELECT LNumber, Year, *Month*, SumOfMoney
FROM T35
WHERE *LNumber = 105961* AND *Year = 1993*

When the user made the selection to view the monthly summary, by use of drill-down operator with the attribute Month, a new form was opened. This form is dynamically defined by the query that is selected from the menu bar. If the user wished to drill-down from this view, another set of arguments for drill-down operator would be given in the menu: {Day, MID}.

Returning to Figure 7, let us choose another menu option. The next query should be generated by using the same condition "*LNumber = 105961* AND *Year = 1993"* but different base table T36 as shown below:

SELECT LNumber, *MID*, Year, SumOfMoney
FROM T36
WHERE *LNumber = 105961* AND *Year = 1993*

This query when executed would find all records with this same LNumber and year and display the results of this query in a new form shown in Fig. 9.

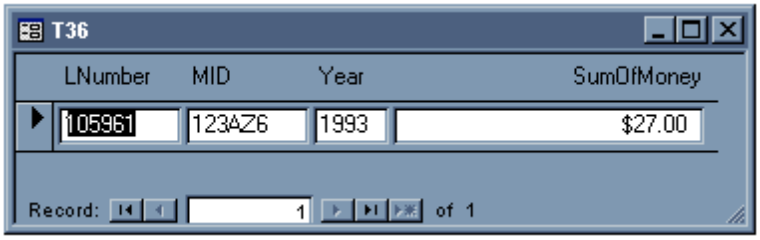


Fig. 9. View of T36.

Using the general aggregation graph, the set of many possible options from any one node, the queries associated with each selection, and the result could be determined.

It is valuable to have the drill-down operator able to drill down on several records as well as just one. Detailing a set of records may be more useful then just one record. By having both options, the flexibility of the operator increases. Selecting multiple records generates SQL statements similar to those used when querying one record. The following examples will also consider the drill down from T43 to T35 and T36 as depicted in Figure 6. The creation of Figure 7 still utilizes Q35. This query will be the basis for which the following queries will be developed. The values of the fields for the records selected will be appended to Q35 as conditions. The following examples will consider that all four records in Figure 7 are selected.

From T43 to T35, the drill-down parameter is Month. The fields LNumber and Year are the two fields defining the conditions for the drill down. Each record selected as a conditional statement, LNumber = "value" AND Year = "value". Therefore, four statements of such a structure will be appended to the second query using the OR operand.

SELECT LNumber, Year, *Month*, SumOfMoney
FROM T35
WHERE *LNumber = 105961* AND *Year = 1993*
OR *LNumber = 105961* AND *Year = 1994*
OR *LNumber = 548741* AND *Year = 1995*
OR *LNumber = 670582* AND *Year = 1999*

Similarly when the drill-down parameter equals MID, viewing the yearly summaries for all machines at the selected location(s) for the selected year(s), the conditional statements are appended to the query using the OR operand.

SELECT LNumber, *MID*, Year, SumOfMoney
FROM T36
WHERE *LNumber = 105961* AND *Year = 1993*
OR *LNumber = 105961* AND *Year = 1994*
OR *LNumber = 548741* AND *Year = 1995*
OR *LNumber = 670582* AND *Year = 1999*

Using the above method of drilling down on several records provides a different view of the data. This functionality makes the operator more flexible.

6. Development of personalization graph

The general drill-down graph, shown in Figure 5, includes all possible views of the data stored in the data warehouse. Personalization of this graph is necessary for several reasons. First of all, not all nodes of the general aggregation graph present useful or logical data groupings. Therefore, those nodes that do not provide a valid detail of the data should not be provided as options for the user to view. Secondly, the wider the scope of possibilities for drilling down, the more complicated the process can get. By confining the user or group within the area of the data warehouse most useful to them, the likelihood of unnecessary and costly processes is reduced. Additionally, security issues can be more systematically addressed within this controlled access.

The general drill-down graph offers many possibilities of personalization. The personalization graph is created on the base of a subset of nodes from the general drill-down graph. Figure 10 shows an example of a personalized graph. It can extend beyond the materialized graph as also shown in Figure 10. T61, T51, and T52 are not included in the materialized graph but are views that can either be created or derived from existing tables. The shaded portion of Figure 10 includes those nodes of the personalization graph that are contained in the materialization graph.

There is another important implication for using personalized drill-down graphs. Analysis of a complete set of personalized drill-down graphs can aid in determining the need for materialization of some views. For example, if many personalized drill-down graphs would contain T52 node then it might be considered to materialize that table. Determination can be done using similar procedures to those described in the literature [Czejdo 2000].

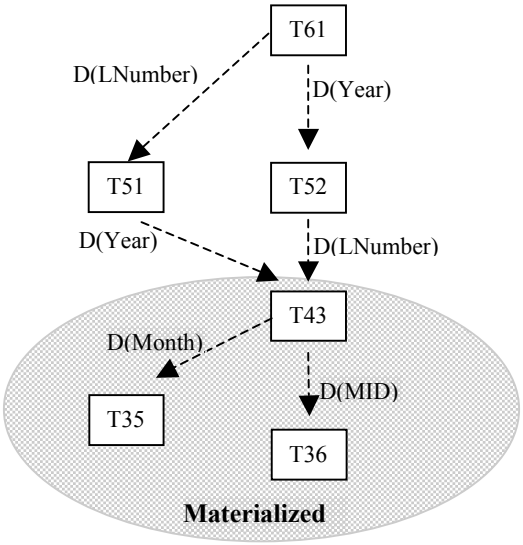


Fig. 10. Personalized Graph

7. Conclusions

In this paper, we considered the problem of generalization of the drill-down operator for an information system. We introduced the concept of a general aggregation graph that is a good basis for defining all options for the drill-down operator. By reducing the scope of the general aggregation graph, a personalized graph, specific to a user(s) is developed. When the personalized graph is examined and the nodes included in its scope are defined, the functionality of the drill-down operator is determined. This graph represents the capabilities of a flexible drill-down operator.

References

Bischoff J. and Alexander T. (1997);*Data Warehouse: Practical Advice from the Experts*. New Jersey: Prentice-Hall, Inc.

Czejdo. B, Taylor M. and Putonti C.,(2000); "Summary Tables in Data Warehouses". Proceedings of ADVIS'2000.

Gupta A., Harinarayan V., and Quass D. (1995); "Aggregate-Query Processing in Data Warehousing Environments", *Proceedings of the VLDB*.

Widom J. (1995); "Research problems in data warehousing", *Proceedings of the 4th Int. Conf. CIKM*.

Section 3:

Knowledge Management Applications

An Agent-Based Decision Support Environment in Collaboration Platforms

Dr. Uwe Forgber, Tim Kalbitzer

conject AG
Munich, Germany

1 Abstract

Today, web-based collaborative systems are mostly limited to the direct support of human actions such as document exchange, communication and process management. However, complex problems do not necessarily have to be solved exclusively by humans but instead by a careful combination and interaction of human and software-based collaborators. This creates high requirements on both the structure of knowledge representation within a given collaborative environment and the in-depth definition of domain specific agents and their underlying business logic. Presently, such prototypical solutions exist for the management of transport corridors, the loading of cargo ships**,** and the design of buildings.

This paper discusses the current development and future vision of conject – a German software service provider focusing on collaboration technology for the AEC (Architecture, Engineering and Construction) and EDA (Electronic Design Automation) industries. The statements and examples are based on conject's product roadmap and market perception that in dynamic markets with shorter product life cycles flexibility and real time decision making are more critical success factors than routine task optimization (Gareis 2003). With an increasing demand on flexibility and more complex products (e.g. buildings or software projects) there is a big need to hide complexity from project participants and achieve shorter ramp-up times.

2 Keywords

web-based collaboration; collaborative agents; mobile collaboration; agent-based decision support systems.

3 Introduction

Today, web-based collaboration systems are mainly focusing on project-based exchange of documents and the traceable communication of geographically-dispersed project participants. Among all industries project owners and participants are just learning to implement and utilize adequate organizational structures and project processes addressing the new opportunities of project collaboration.

However, project collaboration and management still require an enormous amount of tedious tasks and sub tasks (e.g. workflow control, reporting, milestone assessment) to be accomplished by human experts in order to keep the project in time and budget. Process automation – a desired relief – can only be achieved by utilizing primitive workflow engines capable of tracing single user tasks (Schmid and Stanoevska-Slabeva 1998).

As almost all project work is accomplished within a digital environment, process automation, machine based reasoning and decision making can now be easily addressed. Next generation collaboration environments already include basic structures to host domain-specific software agents for process automation and decision support. Based on two examples in the AEC industry this paper outlines the continuing migration process from manually controlled collaborative environments to agent-based decision support platforms within next generation collaboration systems.

4 Collaborative Agents

What are computer-based software agents anyway? Based on their capability to understand any given problem, they commonly collect data and support the reasoning and decision making process in a typically very narrow domain such as matchmaking (e.g. consumer area), command and control or complex, object-centred planning and execution. They can apply domain-specific knowledge on a given problem represented either by a product modelling approach (e.g. object model of a building) or a simple process model representing interdependencies of several tasks over time.

Today, most software agents are embedded in a process model and typically act on a stand-alone basis. Integrated agent environments based on object-based knowledge representation and reasoning are still in their early stages but are rapidly emerging e.g. in scientific and military areas of application (Pohl 2001).

So far, state-of-the-art web-based collaborative environments lack the integration of software agents. However, since their areas of application (AEC, EDA etc.) are subject to high decomposition in practice**,** it is highly inviting to apply software agent concepts in this field. In particular, since user communities of collaboration systems are growing exponential (as line organisations vanish in favour of project-oriented organisations), a wave of automation in process management and decision making can be expected.

5 A product roadmap as a basis for integrating software agents

While a lot of companies in the AEC industry in Germany have been struggling over the past few years, many companies were founded to offer especially to this industry a sound solution for collaboration known as internet-based project management. Compared to the overall developments in this industry, these companies have shown remarkable performance. The success of using collaborative systems for conducting construction projects led to the observable ambition of integrating other existing IT solutions and services into the collaboration platform (e.g. cost control, facility management, CAD (Computer-Aided Design) or reprography services).

In order to become an industry leader, the main challenge for collaboration providers is to develop a platform on which present and future solutions can be hosted. As an example, figure 1 and 2 give an overview of the present products and the architecture for an integration platform of conject, a leading German collaboration provider.

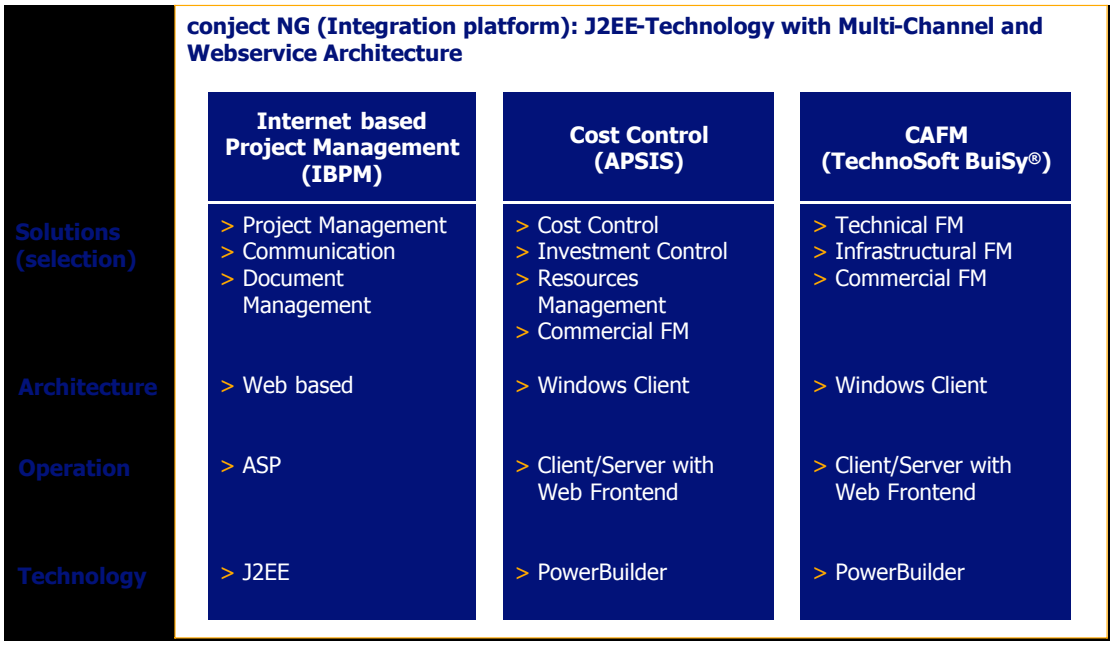


Figure 1: Overview of conject's products (Source: conject, 2004)

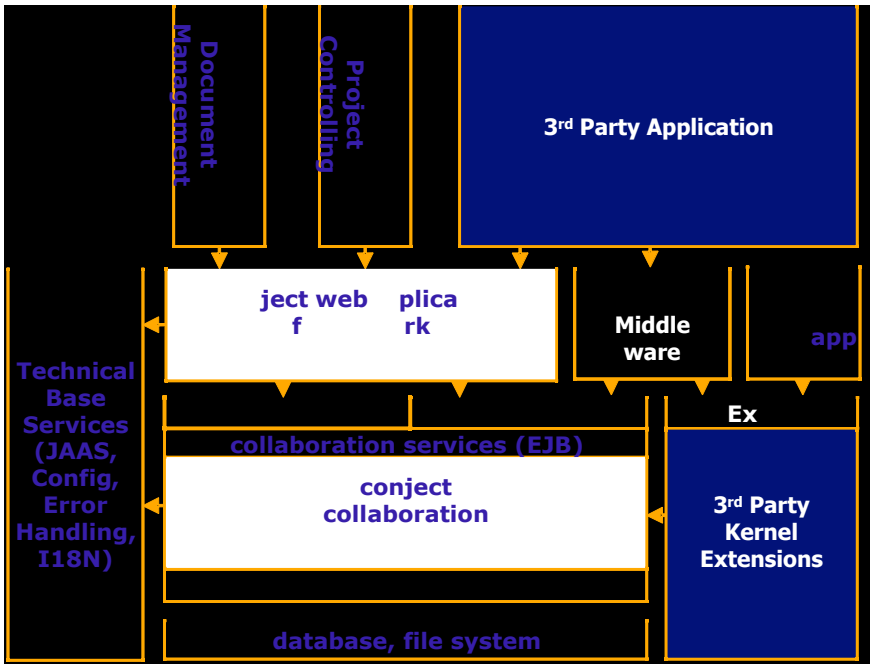


Figure 2: Architecture of conject's next generation product(s) (Source: conject 2004)

The vision of many collaboration providers now is to offer software and consulting services for project-oriented organisations. In dynamic markets companies are forced to be highly flexible and to learn quickly and effectively in order to meet the necessary demands. Consequently, a lot of companies are abandoning traditional business lines and transforming themselves into project-oriented organisations. To successfully transform themselves these companies need organisational change and development on the one hand and a suitable IT system on the other hand.

6 Supporting collaborative work with software agents

Presently, the main advantage of internet-based project management platforms is that all project-related information is stored within one system. This enables project participants to access all project data at any time and from any place. It also facilitates the work of software-based collaborative agents. By adding more information to documents which are relevant for the user document management systems will change and collaboration platforms will evolve from document-based systems to database-based systems. The implementation of software agents helps to reach the next step in process automation and ensures that the desired quality is achieved.

Currently, actions on collaboration platforms are performed mainly by human hand. There is hardly any support by automated processes like process agents. The challenge for software companies designing collaboration systems now is to add intelligence to their future program releases by implementing software-based collaborative agents (Figure 3).

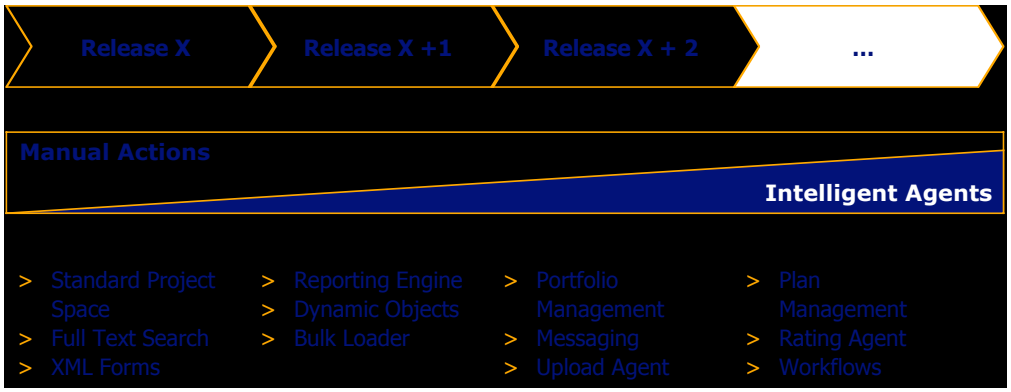


Figure 3: Addition of intelligent agents to future product releases (Source: conject 2004)

A lot of information and processes offered in collaborative environments are static and not addressed specifically to the different user groups. Reports about login times, document access, used space etc. require a high degree of definition efforts and don't offer dynamic analysis possibilities. Workflows which are presently the most sophisticated tools for process automation are also mainly static and immune to ad-hoc changes. A "What's new?" section that just highlights the five latest notifications, mails or events doesn't offer much help to a project participant. It is more relevant to find out what's new for your most important documents and processes.

Simultaneous Work

An important success factor in cross organisational collaboration environments is situation awareness (Marschall and Ackerson 2004). Two project participants not working together in one office often do not know that they are working on the same problem and/or document. Implementing a messenger function that tells both users that they are working on the same document and that both of them are online will offer valuable help. They can immediately contact each other, solve the problem and avoid double efforts. The solution can be provided by two agents who track certain folders and/or documents of their users and compare them with each other. If they match, the simultaneous working users will be informed by an online-message.

Another solution to increase the situation awareness can be displayed by colours. The change of a defined colour spectrum then shows how well the contents of collaboratively working people match. The colours will be displayed in a cockpit-like environment summarising all relevant information.

Reporting
The starting point for evaluating data is collecting it. Event tracking is a collaboration feature that is already widely used. A user-specific profile managed by a software agent can match current events with the user profile and filter the information according to the user's interest. The form of notification will then be chosen according to the user's prioritisation: SMS (Short Messaging Services) for high priority, e-mails for normal priority and messages within the collaboration system at the time of the next login for low priority. By subscribing to certain events of interest and/or configuring search agents all events connected with a document will be summarised using the pull notification concept, i.e. the user will be provided with the relevant information automatically instead of searching for it.

Rating of documents will also help to add valuable information about the content of documents. Agents are then able to determine the value of a document by combining the number of downloads with the given rating.

Workflows
More flexibility will be given to the presently static workflows by more sophisticated IT support. Agents offer a higher automated control process of workflows. If there wasn't any activity from a workflow participant for a certain time notifications will be sent to deputy, supervisor or other workflow participants in order to assure adequate reaction times.

7 Sophisticated solutions for software-based collaborative agents

Project Control
Presently, a software system offers a high amount of different reports to analyse the current project status. The project manager has to generate many of these reports, collect the relevant data and sum everything up in e.g. a new spreadsheet. This requires a lot of administrative work within the system before the real task starts: project control.

A software agent offers valuable help when collecting and showing all the relevant data. Cost data will be collected, aggregated and even visualised in the desired manner. Budgets will be controlled and overruns can be noticed as soon as possible. The big advantage of agents will be the control of a whole project portfolio. Here the analysing process is far more difficult as it contains a much higher complexity. A lot of different people work in different places on different project tasks. It is always the main challenge in project work to use the people involved to as high a capacity as possible (and bearable) and guarantee project success. A sophisticated project agent will find out that a successful project is lacking some resources and will propose solutions like hiring people from different projects that won't be that successful or where the missing of a deadline can be taken into account. In this way, a reporting agent will control earned value developments on a constant basis and prepare relevant information for human decision making.

It is very important to note that the given solutions of a software agent have to be comprehensible for the project manager. Otherwise, the decision support would be a black box that nobody could/would trust. Furthermore, these agents should only be assigned with tasks that they really can handle. There are always so many different basic conditions to consider that can't be fully integrated into an agents mind.

Connexion of reprography services
When adding additional services (e.g. reprography services) to collaboration platforms**,** software agents can also act as brokering agents. According to parameters like maximum/minimum price, number, format, material of plans and location of the company these brokering agents will collect offers of reprography services via SOAP (Simple Object Access Protocol) connections. This increased transparency in the market will decrease the price for blue printing and also help the reprography services to plan their capacity and work more efficiently. If a reprography service uses a brokering agent price negotiations will take place on a digital basis according to the preliminarily determined price and quantity. The collaboration provider simply offers a web front end where participating reprography services just have to enter their bids as in reverse auctions.

8 Technological structure of software-based collaborative agents

The collaboration kernel acts as the "environment" for the software agents. The agents would be JAVA modules using the collaboration API (Application Program Interface) to retrieve the information on which they decide to act. The collaboration API offers various ways for agents to retrieve information. The most important ones are event listeners attached to the currently active user sessions and the reporting engine. An agent listening to a user session would be able to act upon certain user actions or to find out, which users (and maybe their agents) are on-line at a certain point of time. Being implemented as message driven beans, they would be able to communicate with each other on a XML (eXtensible Markup Language)-based data exchange protocol. The agents would be able to use point-to-point communication to other known agents as well as broadcasting channels to post information about their current status or task without knowing whether there are any interested parties.

Agents are not limited to work only within the collaboration kernel. Using SOAP-based web services offered by other systems an agent could have access to external knowledge pools like exchange rates, value added taxes, price of (raw) materials, construction costs etc. Then, information gathered within the collaboration kernel could be compared with external market data or be enriched with additional data from the outside.

In this paper the focus lies on process agents that facilitate and control processes because this will be the next step to integrate agent intelligence to collaboration platforms as described here. One step further would be to design and implement object-based agents. They have a much higher complexity and should be described and analysed at another place.

9 Conclusion

Web-based collaboration systems are currently advancing in several project-oriented industries with an exponential scale. They leave process coordination and decision making entirely to their users. This paper shortly discussed basic concepts and applicable fields of software agent technology and draws the link to web-based collaborative environments.

Based on two examples, the current development of next generation web-based collaboration systems (integrating software agent concepts for process automation and decision support) has been outlined.

Further development of web-based collaboration systems from a task-intensive work bench to a cockpit-like environment for task specification, decision making and execution is therefore anticipated.

10 References

Gareis, R. (2003); Management by projects: Specific strategies, structures, and cultures of the project-oriented company; The Handbook of Managing Projects, Vienna, Austria.

Marschall, M. and D. Ackerson (2004); Building Sustainable Information Systems for Project-Oriented Cross Organizational Collaboration; Proceedings of the Americas Conference on Information Systems (to be published).

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, USA.

Schmid, B. and K. Stanoevska-Slabeva (1998); Knowledge Media: An Innovative Concept and Technology for Knowledge Management in the Information Age; Institute for Media and Communications Management, University of St. Gallen, Switzerland.

Knowledge Sharing, not MetaKnowledge

How to join a collaborative design Process and safely share one's knowledge

Gianfranco Carrara* Antonio Fioravanti* Umberto Nanni°
gianfranco.carrara@uniroma1.it
antonio.fioravanti@uniroma1.it
umberto.nanni@uniroma1.it

* Dipartimento di Architettura e Urbanistica per l'Ingegneria
° Dipartimento di Informatica e Sistemistica "Antonio Ruberti"

Università degli Studi di Roma "La Sapienza"
Via Eudossiana, 18 - 00184 Roma – Italy

Keywords

architectural design; collaborative design; distributed knowledge bases; metaobject; knowledge ownership; knowledge privacy

Abstract

The present paper is drawn from an on-going research on collaborative design, which has been pursued by this Research Unit for a number of years. The proposed model, the resulting system and its implementation refer mainly to architectural and building design in the modes and forms in which it is carried out in advanced design offices. The model is actually used effectively also in other environments. The research simultaneously pursues an integrated model of the:

- structure of the networked architectural design process (operators, activities, phases and resources);
- required knowledge (distributed and functional to the operators and the process phases).

While several aspects of the process structure were illustrated in the previous Symposium (Carrara and Fioravanti 2002), the present article deals essentially with the second point of the model: how the designers, "actors" in the broad sense (according to the ISO-STEP definition, Wix, 1997) in the design process share their own knowledge and how this can be exchanged among them ("Sip by sip knowledge" and the "esperanto interface" XML) in the various stages of its development. As far as the problems involved in the various knowledge bases of the several *actors* exchange are concerned, a short illustration of the proposed solution will be given at the end of the paper.

1. Introduction

In collaborative design support systems the aspects related to the sharing of knowledge and the way in which it is exchanged involve a number of different problems: database structure, homogeneity of the knowledge bases, the creation of knowledge bases (Galle, 1995), the representation of the IT datum (Eastman et al. 1997; Eastman 1998; Papamichael et al. 1996; Kavakli 2001; Rosenmann and Gero 1996; Kim et al. 1997; Pohl, J. et al. 2000; Pohl, K.J. 2002), the ease with which it can be managed, the possibility of rapid growth, etc. Research is part of

the line involving support systems for collaborative design-construction using distributed knowledge bases. Our System, MetaKAAD, is conceived of as an environment in which different Intelligent Assistants, hereinafter denoted as IA, talk to each other. They are functional to the design of a building.

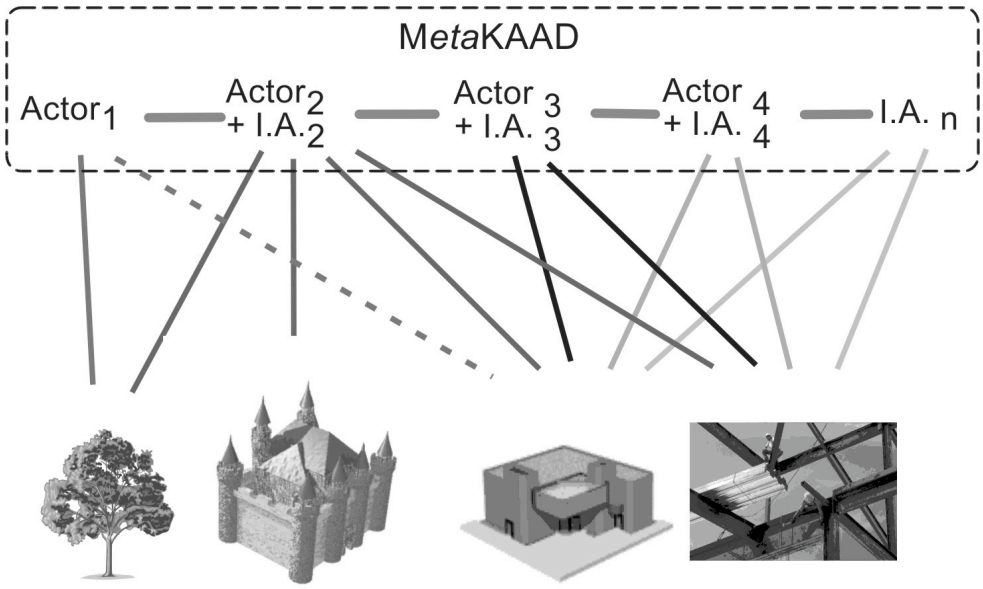


Figure 1. Actors IAs and collaborative architectural design.

The System scheme has been defined in (Carrara and Fioravanti 2001) using knowledge engineering methods and techniques through a structured set of Intelligent Assistants, each element of which is composed of an actual knowledge base (KB), inference engine, assembled database, user interface, graphic primitives (Carrara and Fioravanti 2001). The KB is then composed of a Technological Domain, Space Domain on the one hand (Carrara et al. 1994) and Relation Structures[1] MetaKnowledge, on the other (such as the MetaKnowledge we will see below).

Several IAs are immersed in MetaKAAD, each of which may be structured in a different way according to the actor involved in the phase of design development and to the scientific disciplines involved. It should be noted that in the collaborative design paradigm the various *actors* involved in the design process, although working together closely and jointly on a project and towards its successful outcome, may be in competition with another project taking place simultaneously with the first.

This situation emphasizes the central theme of the article, i.e. the aspects to which little attention has so far been devoted: confidentiality, intellectual property and knowledge security. They display an essential character in a civil society such as the more restricted environment, of collaborative architectural design.

[1] We have defined the Relation Structures in order to take into account the numerous links between the Technological Domain and the Environmental Domain and the rules by which we act on these Domains.

We deal with knowledge for designers, mainly building architects, in which for reasons of professional practice and academic tasks, we are deeply involved. One peculiarity of architectural design resides in the fact of designing single buildings (Eastman, 1999); it may thus be claimed that in most cases it is an activity aimed at construction prototypes rather than actual types. This peculiarity is the result of the dynamic relationship between design objectives/means for attaining them: they influence and modify each other reciprocally (Carrara et al. 1994, pp. 149-150, pp.163-166). This characteristic is gradually spreading through the world of industrial design and the considerations made here can easily be extended to other sectors. One problem facing the *actors* is how to share knowledge with other *actors* in the design process, but without leaving the specific knowledge of their own sector to others. Indeed, owing to the very nature of collaborative design, two conflicting needs must be taken into account: on the one hand, in an immediate and asynchronous way, all the explanations of the conflicts arising in the course of the design and construction process; on the other, safeguarding one's own overall knowledge assets which may be used simultaneously in another project – contract – construction competition in which the previous joint collaborators enter into competition with the new ones. How can this be done?

2. The information exchanged

We envisage possibly tackling these problems via two peculiarities of the software system we are developing. The first of these was illustrated in the preceding Symposium (Carrara and Fioravanti 2002) by defining the relationship PDW/ SDW, between Workspace in Private Design Decisions PWD (inherent in the decisions of the individual design team), and Workspace in the Shared Design Decisions (inherent in the decisions of all the design teams), SDW for *actors* in the design process.

The decisions taken in the "private" design space of a design team or of an *actor* are closely related to the type of support that can be provided by a collaborative design system. For example, pre-set values conforming to the rules, automatic checks performed by activating procedures and methods, the reporting of 'local' conflicts, methods and knowledge for the resolution of 'local' conflicts, the creation of new IT objects/ building components, who the objects must refer to (the 'owner'), the 'situated' aspect (Gero and Reffat 2001) of the IT objects/building components.

2.1. 'Sip by sip Knowledge'

We have defined the second peculiarity as consisting "in providing only the explanations relevant to non-respected constraints and/or the procedures activated without any transfer of parts of the KB from one actor to another: the responses are moved, not the Relation Structures (Carrara and Fioravanti 2001). This modeling may be defined as "Sip by sip knowledge" in the sense that it is dispensed in small doses. The mathematical formalism describing this situation is the small "o", when a function is valid in a limited field, is a "limited scope" knowledge environment.

In the previous symposium the model was conceived of as affording the actors in the design process access to other people's knowledge. Moreover, the formalized knowledge and the logical-mathematical operators required to manage it were included in the other SKBs. This was

done in order to speed up the process, so that on the second request from the same party for knowledge necessary to the process for that actor at that stage was made available locally. This had the primary aim of reducing the number of repetitions of roughly similar information exchanges, and of reducing the bandwidth involved in the transmission. In this procedure little attention was paid to the needs of privacy, copyright and security.

Our position has changed vis-à-vis that illustrated in the preceding conferences. Information exchange now takes place remotely using managers that handle that part of the knowledge that an actor may be interested in.

The new model emerged under the influence of four factors:

- increased demand for security;
- infrastructural development;
- technological progress;
- closer adherence to professional practice.

The first factor is the result of a number of circumstances. The political scenario is now characterized by instability and precariousness, on-line fraud, web site violation, industrial espionage. The guarantee of privacy is a priority need of modern society, not so much in the narrow sense, but as the possibility of using information to construct contexts for presenting it in which its original meaning is modified.

The second refers to innovation, to the IT infrastructures of the territory which is subject not only to expansion but also continuous innovation. Moore's laws predict the doubling of circuit density every 18 months, those of Nielsen the doubling of connection capacity expressed as bandwidth per domestic user every 20 months. Today the PCs on the market are equipped with 1 Gbps network cards and the voice-data backbones are in the order of 10÷20 Gbps. 64 bit processors will replace the previous generation within the next two years.

The above points are gradually introducing a third factor as economic viability, power, the spreading of new apparatus, combined with a strong move towards security have led to the sharing of technologieis such as tunnelling, cryptography, digital signatures, safe internet connections, the development of safer 64 bit S/W and document copyright.

The fourth factor is perhaps the most significant. It is precisely design practice that reveals that the greatest obstacle to open and loyal collaboration between teams and individuals is represented by intellectual property. There is a fear that what has laboriously been learned over time through experience and research can only too easily and immediately be re-transmitted to and acquired by others. This represents an obstacle to increasing the number of actors with whom to collaborate, and design teams tend to be reduced to small groups of collaborators. If it were possible to restrict the transfer of KBs, the number of potential collaborators would increase.
In actual fact, this problem is easier to contain for a number of regions:

- the *actors'* access to information, thanks to the SDW/PDW model, is allowed only through their respective *Perspectives* (Carrara and Fioravanti 2002, p. 33, p. 38);
- modification of the objects is allowed only through the principle of the authority of the objects themselves: *Dominus*, *Owner*, *User* (Carrara and Fioravanti 2002, pp. 39-41);
- not all the information exchanged represents knowledge;

- the information physically shared by the actors is not sufficient to infer knowledge from.

The first two points were dealt with in the preceding Symposium. Here we shall examine the third point. The last one will be illustrated in a subsequent section.

2. 2. Data Knowledge MetaKnowledge

The information exchanged may be catalogued in four sub-sets: Data, Knowledge, MetaKnowledge.

The first subset, *Data*, consists of instances of the prototype objects accessible to all the actors involved in the project in accordance with their respective *Perspectives*, but are limited to describing the *building entities*[2] referred to by the various actors. There is no transfer of knowledge, only dimensional, physical, economic characteristics, etc. There is no explanation, for example, of why the choice fell on one particular type of door or why it is difficult to relate one type of window frame to a certain type of building facade.

The second subset, *knowledge*, at least as we understand it, refers to a more limited knowledge environment than that exchanged for the *building entity* considered. Examples of this are: the dimensional range of construction feasibility of a type of prefabricated wall pane; incompatibility between the position of beams and air conditioning conduits in one particular point in the building; an incorrect ratio between window surfaces and illuminated room areas dependent also on the orientation of the outer surface of the building with respect to the heliothermal axis and its inclination to the vertical; the ratio between the thickness of the slab and its free span with a certain type of slab and boundary constraints; control over the minimum distance to travel towards the first safety exit. This type of information always resides in the IT objects representing the building entities, both as functions and as procedural attachments, in the prototype object or in its parent prototype or in a prototype of which it is part. As shown, the exchanged knowledge refers only to the explanations, the instances thereof, for a given "situation" or "context", not the constraints that validate the *Requirements* and that remain in the KB prototypes of the respective *actors* (Carrara and Fioravanti 2003).

The third subset, the M*eta*Knowledge, refers to a larger PDW knowledge environment than the previous one and refers to the rules applied to the previously illustrated norms. Here the boundary conditions of the state of the System are extremely important. They make one design choice preferable to another. The implementation of the peculiarities of each event and its knock-on effects and relations to the other objects enrich this part of the KBs which we have defined as Relation Structures. Indeed they structure the relations among the objects presents in the Environmental Domain and in the Technological Domain (Carrara and Fioravanti 2001). This is true only in restricted environments. Indeed it is only the boundary conditions, albeit in a wider environment, that make one solution preferable to another, and these boundary conditions, which represent the peculiarity of each situation and that go to make up M*eta*Knowledge, are not

[2] The *Building Entities* represent a huge field of prototype objects: a building *Component*, a *Relation Structure* among objects, a higher-level *Prototype*, an *assembly sequence*, a *technical element*, an *entire building*, an *activity*, a *constraint*, a *method*.

transmissible because we must take into account all the events that, when tested against experience, will lead to a choice.

3. The Proposed Model

In developed systems we consider there are several KB_i that act in a set mode (fig. 2) (Carrara and Fioravanti 2001, §3.2).

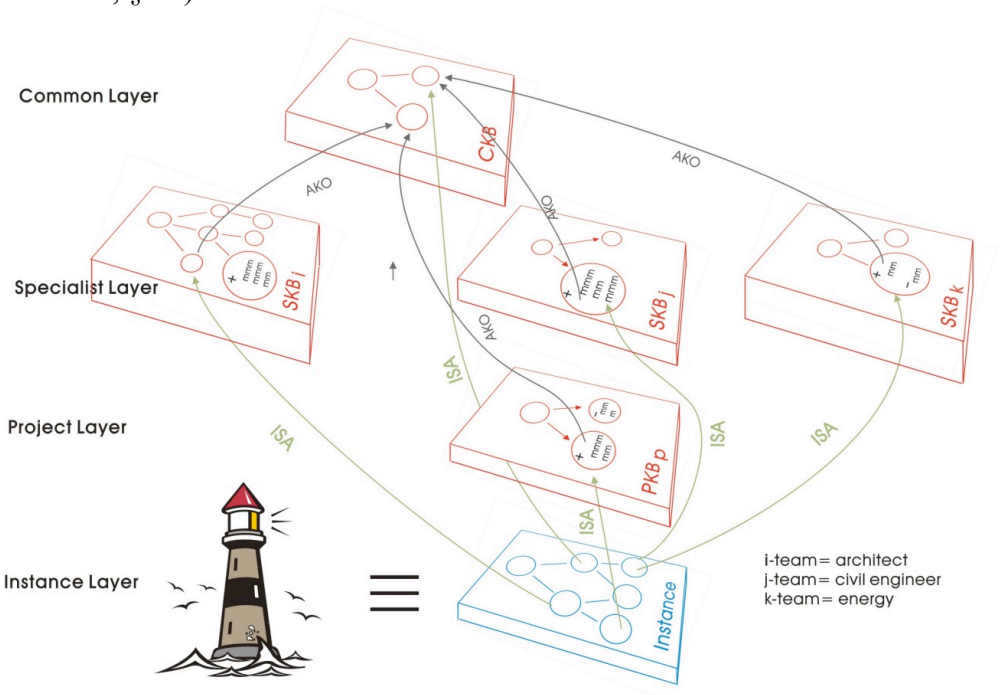


Figure 2. The knowledge bases and the entire building.

The running Knowledge Base union set for designing the *Entire Building* (EB) is constituted by:

$$\square\ KB = CKB + \square\ SKB_i + PKB_p\ \square \qquad (1)$$

$$\square\ \bigcup_{i=1,n}^{p}(KB\ i,p) \bigcup CKB = \bigcup_{i=1,n}^{p} KB\ i,p,c$$

where:
{i □ Actors | i = 1, n; n variable □ Phases □ Specializations}
CKB = Common Knowledge Base of all actors;
SKB_i = Specialist Knowledge Base, peculiar to the i-actor;
PKB_p = Project Knowledge Base of the p-project.

It is important to stress that no hypothesis was made about the implementation and the structure of SKBs, as we think that each *actor* develops the representation most suitable for his own problems.

An *actor* in the process, in order to instance any kind of IT object, after a first verification phase within his own PDW and the respect of any constraints present, and later goes on to the second phase of verification, the TEST phase in the SDW, concerning other SKBs. In order to perform

such a verification in the previous system, the author would have to add to his knowledge the knowledge of the new *Requirements* (Carrara and Fioravanti 2003). Hence new characteristics would be added and multiplied set-wise in the IT objects of his own SKB_i (Carrara and Fioravanti 2002). By so doing however he would acquire the Knowledge of the other actors and thus come into possession of the intellectual property of others.

From the point of view of the actor modeling knowledge exchanges take place according to the following scheme: its i-th Specialist Knowledge Base Base evolves in the course of the project, increasing in t-th time its knowledge in (t-1)-th time by the knowledge of other actors in (t-1)-th time. This acquired knowledge obviously refers to the characteristics of the objects enucleated by the intersection sets between its own and the other SKB_i (see equation (2)); i.e., the objects of other SKBs that are not the complement of their i*-th SKB (see equation (3)).

The technique involved in this problem was similar to the one adopted by Rosenman and Gero (Rosenman and Gero, 1997).

$$(SKB_{i*})_t = (SKB_{i*})_{t-1} \cup \circ(SKB_{i*})_{t-1} \bigcap_{i=1,\, i\neq i*}^{i=n} (SKB_i)_{t-1} \quad (2)$$

$$= (SKB_{i*})_{t-1} \cup \circ - \complement (SKB_{i*})_{t-1} \bigcup_{i=1, i\neq i*}^{i=n} (SKB_i)_{t-1} \quad (3)$$

where:
{i ∈ Actors | i = 1, n; n variable ∈ Phases ∈ Specializations}
$(SKB_i)_t$ = Specialist KB, peculiar of i-th *actor*, at t-th time.
$\complement (SKB_i)_t$ = Complement of Specialist KB, peculiar of i-th *actor*, at t-th time.

But now the acquisition is only "virtual", also in view of the increased speed and availability of the processing environments, we are defining a System in which control of constraints takes place in the TEST phase in the others' IAs and KBs, as the first actor sends to them only the data he has defined.

After verifying these data at a distance, the other KBs will provide an explanation of why the proposed solution was accepted or not in that particular situation, providing the rules applied.
In brief, we state that the key concept is to apply and notify knowledge for the single project only for the instance, i.e. to give the answer for the (single) state of problem of that particular instance of the "world", SWD, being dealt with by the actors. In this way no elements are supplied concerning the general rules governing one's own knowledge, the SKB_i.
The data actors can access, and that are actually shared, the Assembled Data Base, are simple structured lists: nothing else is embedded.

We are thinking of extending the preceding concept also on the basis of the project knowledge base, PKB_p bringing it inside the SKB_i and "virtualizing it". Indeed, we must bear in mind how thesystem structure Fig. 3 (Carrara and Fioravanti 2001), now that is has changed, is no longer conceived of an entity in its own right, physically and logically associated with a project to which the actors have access, but is a "virtual" PKB_p, always associated with the project but resident, as far as competence is concerned, in the various SKBs of the *actors* involved. In the present version

it is composed of objects, the features of which are partially modifiable by the other *actors* by means of editing filtered through the Perspectives and the *Principle of Authority*. Hence, in a manner similar to equations (2) and (3) it can be stated:

$$(PKB_{i^*,\,p})_t = (PKB_{i^*,\,p})_{t-1} \cup \circ(PKB_{i^*,\,p})_{t-1} \cap_{i=1,\,i\neq i^*}^{i=n} (PKB_{i,\,p})_{t-1} \quad (4)$$

$$= (PKB_{i^*,\,p})_{t-1} \cup \circ - \complement (PKB_{i^*,\,p})_{t-1} \cup_{i=1,\,i\neq i^*}^{i=n} (PKB_{i,\,p})_{t-1} \quad (5)$$

where:
$\{i \in \text{Actor} \mid i = 1, n;\ n \text{ variable} \in \text{Phases} \in \text{Specializations}\}$
$(PKB_{i,\,p})_t$ = Project KB, peculiar of i-th *actor*, of p-th project, in t-th time.
$\complement (PKB_i)_t$ = Complement of Project KB, peculiar of i-th *actor*, at t-th time.

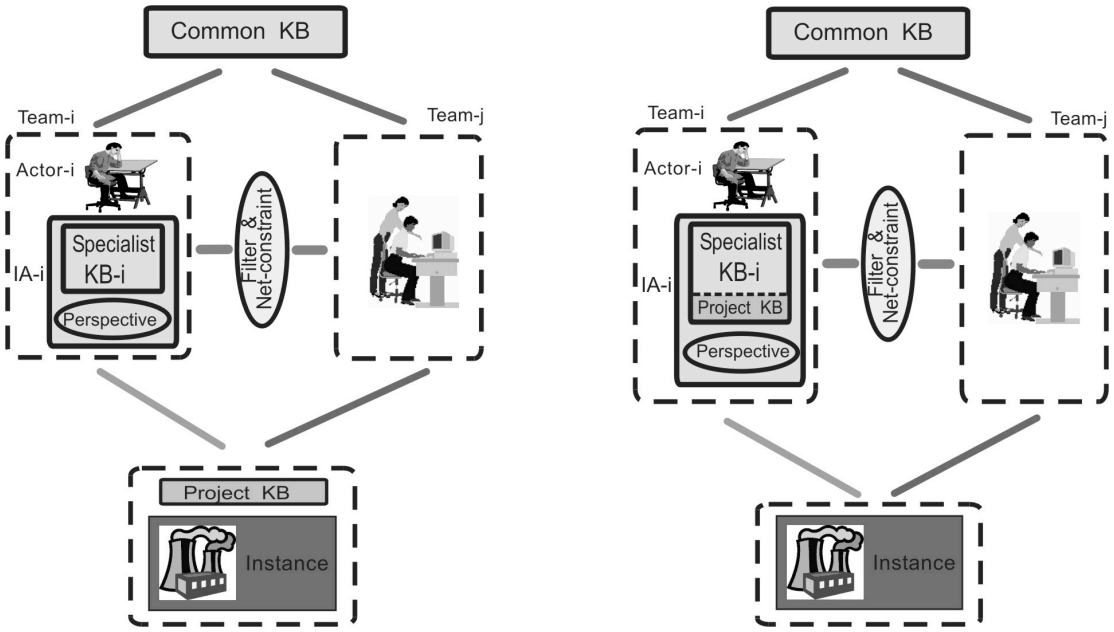


Figure 3. The structure of IAs and of the EB: different PKB conception in 2001 on the left, and current one on the right.

4. Architecture of the Proposed Software: MetaKAAD

As stated in the previous sections, we aim at satisfying a general goal of Collaborative Architectural Design: to integrate the efforts of several actors to produce a project. Each actor has a specific role and responsibility in the design process, and pursues his/her specific tasks based on specific roles, expertise and knowledge.

Within this context, we have identified as a target of the research the development of *MetaKAAD*, a software architecture dealing with these main issues:

- *roles*: each actor has a role, focus, responsibilities and goals within the project;

- *interoperability*: each actor involved in the design process has his/her techniques and may use specific software tools; we want all of them to cooperate in producing the same project, instead of working on unshared incompatible files;
- *consistency*: each actor provides a contribution to the design process by either creating new objects within a project, or modifying or adding "features" (or slot-value pairs) to existing objects; nevertheless the resulting project may evolve consistently while concurrent updates are carried out;
- *synchronization*: there are synchronization rules among the activities carried out by the actors (e.g., you may not want to design a building without electric cabling, but usually you will not start the design process from it); on the other hand, when a parallel activity is allowed , we want that more actors develop their design goals in the same time;
- *communication*: roles and synchronization rules require communications among the actors to be generated and managed.

Roles and *synchronization* rules may change according to the type of project (e.g., you may really plan to start a project from the cabling requirements). This may happen for all the instance-project of a given prototype-project. Hence, we want to model at the level of prototype-project the synchronization rules of all instances of this kind of projects.

The second of these issues, *interoperability*, raises a common problem in the professional practice of architectural and building design: any expert wants to use his/her own design tool. Tackling this problem is a real challenge, due to the different features supported by the design tools of the various actors. These different representations (sometimes a consequence of intentional choices of software companies) are often difficult to override, and constitute a practical obstacle towards the goal of building an actual collaborative design environment.

The *consistency* issue has difficulties similar to those arising in the field of databases, dealing with concurrent transactions. Therefore, our choice is to handle the possible concurrent updates performed by different actors by means of techniques similar to those used in that area. Namely, we aim at enforcing synchronization and avoid critical updates by means of implicit locking of objects and features, based on the concept of *principle of authority*. Implicit means that the actor must not explicitly handle this issue, which is handled by the tool. In turn, ownership of features and objects depend on *Perspectives* defined in the specific prototype-project.

4. 1. The Actor's Role and the "Perspective"

Each actor involved in the design process has the management of a set of features of the project (as a special case, the management of some feature may be shared by two or more actors). Furthermore an actor is interested in some other features. Each actor has an associated *Perspective*, which includes two sets of objects/features:

- those of which he/she is the *Dominus*
- those of which he/she is the *User*.

On the other side, we can look at the *Perspective* system as consisting of two functions:

- *management*: mapping each object/feature to a set of actors (very often, one actor);
- *interested_in*: mapping each object/feature to a set of actors.

Of course, from a practical point of view, the features included in a *Perspective* are dependent on the design tools the specific actor is willing to use. This is a powerful mechanism to progressively extend more and more software tools in the M*eta*KAAD architecture, provided that the "drivers" are built. An initial effort in this direction has been started.

5. Research lines and current M*eta*KAAD XML filters

The project is stored/exchanged using a representation as a XML file: this approach is widely accepted in any area, and common in the context of architectural design, and has produced worldwide standardization efforts (e.g., *aecXML*, *IFC*) by many organizations (Harold et al. 2002). On the other hand our approach is robust toward the changes of the model (i.e. the chosen XML representation format), since these do not affect the key features of the system (e.g. synchronization).

In the Common Knowledge Base, there exists a general XML Data Type Definition (DTD) for each kind of building, i.e., for each project-prototype. This includes:

- structural properties: the essential features of these models are based on the object-oriented models within KAAD[3];
- the roles and *Perspective*s within the prototype;
- a representation of the synchronization rules.

These models are integrated by adding new features from other software tools, used by experts in complementary fields: at the moment we are collaborating with Civil Engineers.

A project consists of a XML file, available in the "shared" design space. This is an instance of a given project-prototype, and incorporates objects and features produced by several actors. When an actor imports a given project into his/her workspace, the global XML file is "filtered" by his *Perspective*. In other words, only the features that are managed by or of interest in the corresponding role in the design process are activated.

After an actor has modified a project with his/her preferred software tool, a new version of the project (i.e., a new XML file) is exported into the SDW area, and the other actors may import the new version.

As a first step toward these goals, in the current prototype of *MetaKAAD*, we focused mainly on the *interoperability* step, consisting in modeling and exchanging projects among various actors, each using a specific software tool.

The interoperability among tools operating – in principle – toward a convergent design target have at least two distinct issues:

- semantics: the tools actually cope with substantially different concepts
- format: different choices in the representation can make it impossible to exchange files between two software tools, even if both essentially manage the same concepts.

We have dealt with both facets of the problem. Let us consider a simplistic (but quite general) scenario, where two actors in collaborative design over the same project P use two design tools:

[3] An existing software system developed at Department DAU (Carrara and Fioravanti 1995).

an architect, Arch, and a civil engineer, CEng, respectively using the tools T_{Arch} and T_{CEng}, fig.4. Each of the design tool T_i (from now on T_i can be either T_{Arch} or T_{CEng}) has an internal representation (semantics and format); this must be used when a file is stored and possibly exchanged across the network between different *Actors*, i.e., between different installation of T_i. From a semantic point of view, the representation P_i contains only the features of the *Perspective*-a.

Let us call P_{Arch} and P_{CEng} (or, in general, P_i) the internal representations of the same project P inside the tools T_{Arch} and T_{CEng} respectively. Furthermore, there exists a XML schema (or DTD) for each of the tool T_{Arch} and T_{CEng} : DTD_{Arch} and DTD_{CEng}, respectively.

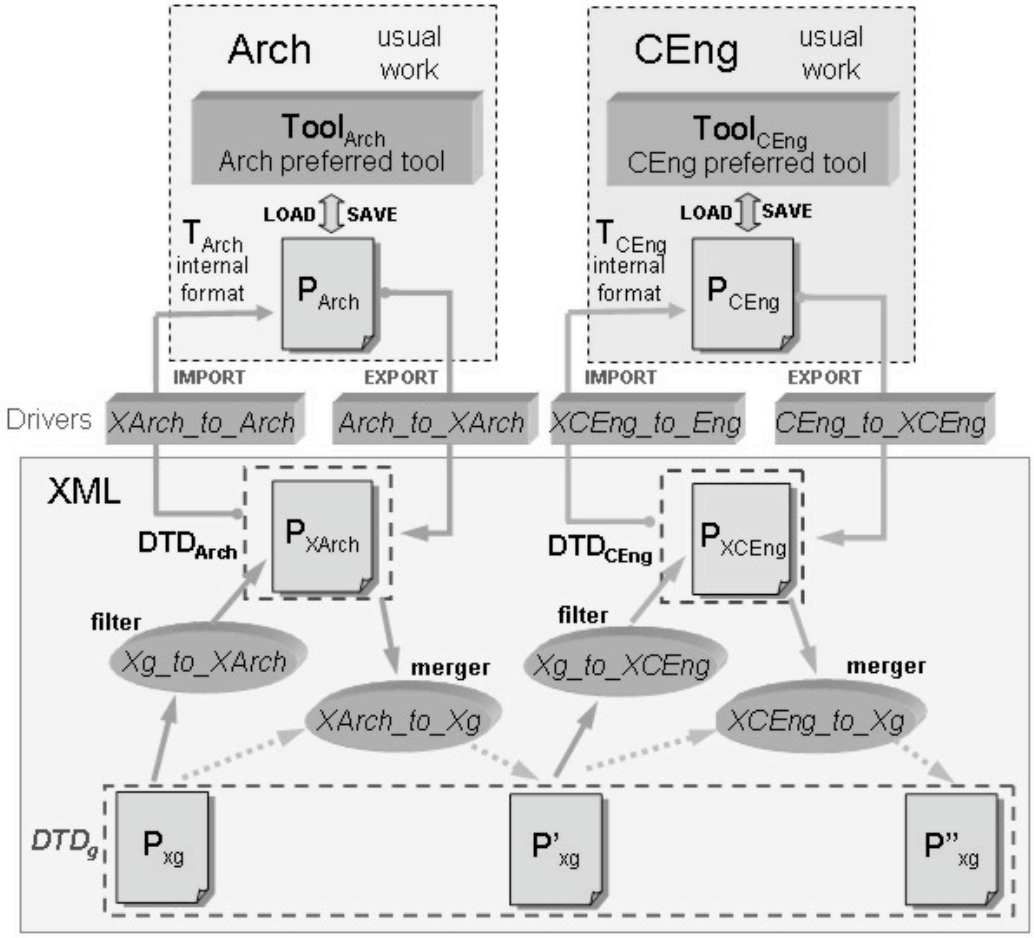


Figure 4. Data transformations within MetaKAAD.

When the architect exports its project P_{Arch} by means of the tool T_{Arch}, it is transformed in P_{XArch}: this is an XML representation using the XML format DTD_{Arch}. To export/import files from/to the tool T_{Arch} , we need two "drivers": *Arch_to_XArch*, and *XArch_to_Arch*. Note that these drivers operates a *syntactic* transformation: all the information stored in P_{Arch} is also in P_{XArch} and

viceversa. The same applies to the civil engineer, using the drivers *CEng_to_XCEng*, and *XCEng_to_CEng* to switch between the files P_{CEng} and P_{XCEng}.

There exist a global XML schema (DTD_g) encompassing both DTD_{Arch} and DTD_{CEng}. More precisely, an XML file which is valid for DTD_{Arch} and DTD_{CEng} is valid also for the global schema DTD_g.

A global representation of a project P_{Xg} can be mapped to P_{XArch} by means of a "filter" *Xg_to_XArch*: this is a *semantic* transformation, consisting in the extraction of a Perspective from a global representation. On the other side, we need a "semantic merger" *XArch_in_Xg*: this more difficult task requires to merge P_{XArch} in P_{Xg} and generate a new version of the project P'_{Xg}. The basic rule of this task is the following:

- for all the features (slot value pair) in the *Perspective*-a which are changed from P_{Xg} to P_{XArch}, replace the value in P_{Xg} with the corresponding one in P_{XArch}.

As a first step to achieve this target, we have developed an XML representation of the project, and have partially developed the mapping procedures (drivers) for very simple and basic structures (semantically shared by the two environments) from/to different target software environments:

- *KAAD* [4]
- AutoCAD.

We also are cooperating with another research group in structural engineering, and have started the connection to M*eta*KAAD of the system SISA, a prototype software tool to design and define load-bearing structures of a simple residential building.

5.1 Additional Features of the System

Beside those considered before, additional features that have been achieved are that each actor, while importing from the shared area his/her own *Perspective*, can easily check:

- which objects/features have been changed, and which are managed by whom
- none of the features managed by himself have been changed by others

Also, the approach based on features and *Perspective*s may lead to the concept of (digital) signature of the *Perspective* of a project, where each actor may digitally sign only the set of features he/she is responsible for, and recognize as unchanged all the features in his/her *Perspective* even after other actors has changed the project.

6. Conclusions

The paper illustrates how most of the obstacles, intellectual property (copyright) in particular, standing in the way of the spread of the collaborative design paradigm, can be overcome, at least in theory. The objection to the access by others to one's own KB is a false problem in that all the actors query directly only the shared data, structured by XML. Knowledge is exchanged through

[4] See note 3 on page 114.

specific *Perspectives* restricting the field of action on the objects by specific actors - *Dominus* and *User* – which are variable in time. The knowledge notified is only a "restricted environment" of SKB_i of the actor(s), which are insufficient for inferring open environments (metarules). Furthermore, it is queried through the *Perspectives* which virtualize the inclusion of parts of KBs in other KBs.

The M*eta*Knowledge of each actor is encapsulated in Relation Structures separated from the KBs of the Environmental and Technological Domains so that direct access to them can be excluded. The signature of each design *Perspective* allows the choices made by the various actors with regard to the tasks assigned to them to be certified. This same digital signature signifies the impossibility of tampering with the work performed, ensuring that the actors involved take responsibility for their work.

This is the state of research as confirmed by the results of the degree theses presented and by the prototype developed. The experience gained through the evolution and reflections of the M*eta*KAAD system presented in the preceding Symposium, leaves several lines of development open.

General conclusions may be drawn to the effect that in this way Knowledge and not MetaKnowledge is shared, "rules of good building practice" is shared and not rules that define when and how to apply the "rules of good building practice".

Acknowledgments

The research was funded by MIUR (Ministry of Education, the University and Research): Project of Research of National Interest 2002: "An integrated Product/Process model of support to collaborative design in building".

References

Carrara, G. and Fioravanti, A.: 2003, Needs Requirements Performances Vs Goals Constraints Values, *in* Collaborative Architectural Design, Proceedings of SIGraDi 2003 Conference, Rosario.

Carrara G. and Fioravanti A.: 2002, 'Private Space' and 'Shared Space' Dialectics in Collaborative Architectural Design, *Collaborative Decision-Support Systems*, Focus Symposium and InterSymp-2002 Conference Proceedings, Baden-Baden, pp. 27-44.

Carrara, G. and Fioravanti, A.: 2001, A Theoretical model of shared distributed Knowledge bases for Collaborative Architectural Design, SKCF '01 Conference Proceedings, 17-18 Dec. 2001, Sydney.

Carrara, G, Fioravanti, A, Novembri, G: 2001, Knowledge-based System to Support Architectural Design, *in* H Penttila (ed.), *Architectural InformationManagement*, Proceedings of eCAADe 01 Conference, Helsinki, pp. 80-85.

Carrara, G, Fioravanti, A, and Novembri, G: 1997, An Intelligent Assistant for architectural design studio, eCAADe 1997 Conference, Wien, a Web page *in* http://info.tuwien.ac.at/ecaade/proc/carrara/carrara.htm

Carrara, G., Confessore G., A. Fioravanti, G. Novembri. 1995, "Multimedia and Knowledge-based Computer-aided Architectural", *in* B. Colajanni and G. Pellitteri (eds), *Multimedia and Architectural Disciplines*, Proceedings of the 13th eCAADe Conference, Palermo, Italy.

Carrara, G, and Kalay, YE: 1994, Past, present, future: process and Knowledge in Architectural Design, *in* G Carrara and YE Kalay (eds.), *Knowledge-Based Computer-Aided Architectural Design*, Elsevier Science Publishers B.V., Amsterdam, pp. v-vii, 147-201, and 389-396.

Eastman, C.M.: 1998, Editorial, Automation in Construction, **7**(6), 431-432.

Eastman C.M., Jeng, T.S., and Chowdbury, R.: 1997, Integration of Design Application with Building Models, in R. Junge (ed.), CAAD Futures 1997, pp. 45-59.

Galle P.: 1995, Towards Integrated, 'Intelligent', and Compliant Computer Modeling of Buildings, *Automation in Construction* **4**(3), 189-211.

Gero, J. S. and Reffat R. M.: 2001, Multiple representation as platform for situated learning systems in designing, *Knowledge-Based Systems*, **14**(7), 337-351.

Harold, R.R.E. and Means W.S. 2002. *XML in a Nutshell*, O'Reilly & Associates Inc.

Kavakli, M.: 2001, NoDes:kNOwledge-based modeling for detailed DESign process – from analysis to implementation, *Automation in Construction*, **10**(4), 399-416.

Kim, I., Liebich, T. and Maver, T.: 1997, Managing design data in an integrated CAAD environment: a product model approach, *Automation in Construction,* **7**(1), 35-53.

Papamichael K., La Porta, J., Chauvet, H., Collins, D., Trzcinski, T., Thorpe, J. and Selkowitz, S.: 1996, The Building Design Advisor, *Design Computation: Collaboration, Reasoning, Pedagogy*, Proceedings of ACADIA '96 Conference, Tucson, Arizona, pp. 85-97.

Pohl, J., Chapman, A., Pohl, K.J.: 2000, Computer-Aided Design Systems for the 21st Century: Some Design Guidelines, 5th International Conference on Design and Decision-Support Systems for Architecture and Urban Planning, Nijkerk, The Netherlands, August 22-25.

Pohl, K.J.: 2002, The Underlying Design Principles of the ICDM Development Toolkit, *in Collaborative Decision-Support Systems*, Focus Symposium and InterSymp-2002 Conference Proceedings, Baden-Baden, pp. 51-58.

Rosenman, M A., and Gero, J S.: 1996, Modelling multiple *Perspective*s of design objects in a collaborative CAD environment, Gero, J.S., (guest ed.), Special Issue: Artificial Intelligence in Computer-Aided Design, *Computer Aided Design*, **28**(3), 193-205.

Wix, J.: 1997, ISO 10303 Part 106, BCCM (Building Construction Core Model) /T200 draft.

Knowledge Management and Organizational Memory in CAS Environments

A. Killing

K+H Architects, Stuttgart, Germany

a.killing@kh-architekten.de

Abstract

This paper addresses aspects of knowledge management and organizational memory within a complex adaptive system (CAS). The increased connectivity and complexity within such an environment creates an open system that displays rarely clear cause-effect-relationships. Due to this characteristic it becomes increasingly difficult to maintain accurate knowledge and representation of the system's properties and behavior generating mechanisms. Considering the dynamic change of the environment an organization is operating in, the paper questions the long-term validity of knowledge that is based on the assumption of a closed system, i.e., clear cause–effect links. The author draws attention to the basic prerequisites of applicable data interpretation and how these correspond to a CAS-environment.

This paper suggests that organizational memory needs to adjust the way it stores and represents knowledge that was generated from past experiences. Attention is drawn to the fact that interpretation of information is closely linked to human factors, as knowledge and perception of the interpreting individual, or even hard-coded rules and relationships within software applications. As results of the interpretation process may vary significantly under dynamic change of these factors, the paper proposes a knowledge management approach that allows simultaneous representation of two or more conflicting interpretations of one set of given information and describes its possible impact on the structure of organizational memory and decision making. The paper concludes with an outlook on limitations and assets of a behavior based approach to organizational memory structures.

Keywords

organizational memory, knowledge management, complex adaptive systems, analogy making, pattern recognition, implicit and explicit knowledge

Introduction – Definitions and Aspects of Organizational Memory

"Those who cannot remember the past are condemned to repeat it"

(George Santayana)

Reduced to its basic essence, the well-known game "memory" requires the human brain to interpret the picture on the flipped card as an analogy to a picture that has been seen before in another place and to remember, where this place was. Unfortunately, real life is a little bit more challenging. Since identical situation only rarely occur, analogy-making requires to reduce the essence of an occurring situation to a set of key characteristics. This is the point where memory becomes a difficult subject. It implies that besides remembering the past, an essential part is to perceive two non-identical objects (or situations) as being the same at some abstract level. The decision, which characteristic of the content is considered to be worthwhile to be represented on an abstract level is closely linked to the intellectual capabilities of the perceiving individual.

This insight may lead to the question what memory really is. Is it the place where information is stored (like a CD-ROM), or is it the ability to store and meaningfully retrieve information and relationship later. Translated to IT-terminology the question may be: Is memory software or hardware. In order to identify different approaches to organizational memory, it is essential to define and describe what exactly this memory is going to store and process: the organizational knowledge.

Aspects of Organizational Knowledge

Osterloh and Wübker (1999) distinguish to types of knowledge: Explicit knowledge and implicit knowledge. To understand these to types of knowledge and their impact in an organizational context, it important to define key characteristics by drawing analogies to different levels of knowledge the human brain deals with in daily life.

Explicit knowledge can be represented in written form. It is easy to copy and distribute. Therefore, it is easy to store and easy accessible. On the other side, this type of knowledge is static and closely linked to given set of surrounding conditions under which this knowledge is valid. If these conditions change, explicit knowledge has to be updated and maintained. Translated to daily life's challenges for a human brain, explicit knowledge may be the ability to use a certain version of a software application. This type of knowledge is easy to represent in a manual. However, if the next version of the application is available, the capability of using this software has to be updated. In an organizational context, this type of knowledge is usually provided by manuals, guidelines and written statements. It also is easy to process with information technology.

In contrast, implicit knowledge is provided by individual skills and experience. The key characteristic of this type of knowledge is, that it is hard to represent and store. The structure is

strongly connected to the cognitive rules of the perceiving individual. On the other hand, by its nature, implicit knowledge is subject of constant change. Since the perception adapts constantly to new insights, the resulting knowledge is hard to represent. Using the aforementioned example of knowledge in software applications, implicit knowledge may enable the human brain to intellectually understand structures by analyzing repetitive occurring patterns in, lets say, the user interface of several software applications. In an organizational context, this type of knowledge represents a high value, because the resulting competence cannot be imitated easily. Moreover, it belongs inherently to the people, who work for an organization.

Current Developments in Organizational Memory and Knowledge Management

In order to minimize the risk that essential parts of the organizational knowledge are linked to individuals, organizations seek for methods to transform this value to explicit knowledge and store it where it is long-term accessible. Organizational memory systems (OMS) used to be focused on the development of tools that will enable massive information transfer within an organization. Since these systems face the problem of availability of the right information at the right time in the right place, representing context occurred as a crucial issue that facilitates appropriate information retrieval and understanding (Conklin, 2001). This approach faces difficulties, because OMSs require additional documentation effort with no clear short-term benefit, and often do not provide an effective structure to the mass of information collected in the system (Boy, 2001).

Referring to the earlier identified types of organizational knowledge, the aforementioned tools only provide the storage and display of explicit knowledge. The representation and management of implicit knowledge mainly relies on strategies that aim on effectively developing and utilizing the human capital in an organization. Pohl (2003) has described, that decentralization and concurrency represent principal characteristics of knowledge management within an organization and emphasizes the goal of creating an environment that builds relationships for the purpose of maximizing interaction, diversity, responsiveness, and flexibility.

Complex Adaptive System Environments as Challenge for Organizational Knowledge

A Complex Adaptive System environment rarely displays the linearity that is required to maintain an applicable knowledge base. Moreover, the non-linear pattern of behavior reduces both the predictability of the system and the manageability of accurate knowledge. Mainly, this property is due to the fact, that our environment the compounding parts itself are open systems, that constantly interact and react to even small incentives and impacts. Self-organization and constant adaptation diminish the ability to retrieve and store easily applicable organizational knowledge. Specific knowledge about the surrounding environment is hardly maintainable on an up-to-date level, because significant details or relationship may change in high frequency.

Limitations of Individual and Organizational Knowledge

The capabilities, to recognize pattern in an constantly changing environment are dependent on the individual history and experience of an individual, a group, or an organization. An organization that employs feedback mechanisms to adjust its performance generates its identity through the history it has experienced (Stacey, 1995). Consequently, this historical path impacts the way of creating analogies to past experience. This insight offers two important statements. (1) Organizations have to develop an organizational memory that provides a holistic base for this process of pattern recognition. (2) The choice of key parameters that describe patterns is based on the individual history of the organization.

This again means that the same event, process, or condition can be perceived and memorized in different ways. Complex adaptive systems employ mechanisms of pattern recognition to anticipate the future (Pascale, 1999). In contrast, Logic deduction is considered to have only limited representation capabilities because it implies the existence of a closed system and clear cause-effect links. Consequently, systems have to employ heuristic methods that are closer to intuitive reasoning than to deterministic deduction. The occurrence of recognizable patterns produces probabilistic assumptions rather than deterministic predictions about the future. In an organizational context, pattern categories may include customer patterns, product patterns, organizational patterns, and mega patterns (Slywotzky, Mundt and Quella, 1999).

Recognizing the ambiguity of perception is essential because it explains why pattern recognition can hardly deliver deterministic results: Different the different ways of memorizing an event react to different analogies. Consequently, an organization has to provide a frequently executed feedback mechanism to both support and monitor its pattern recognition capability. Finally, efforts in creating an organizational memory have to admit that perception, recognition, and reasoning can only produce probabilistic results with limited long-term validity.

Ontology: Relationship as Key Characteristic in Representation

In order to achieve a higher level of understanding with some predictive capabilities, it is necessary to create models that are built to imitate the way of functioning of the real world environment in a manageable scale. To virtually represent the real world environment in a computer based model, information structures have to consider not only the objects an their properties. They also must display relationships in-between the single elements of a system. This type of information structure consists of software agents that interact with other agents on the base of hard-coded rules. Since a system is directed rather by relationships than by characteristics of its elements, an ontological model is able to display system dynamics that emerge from the complex property of a system. However, this approach also implies that all relationships within the system are known and can be described by rules. External relationships that may also impact the system behavior may be neglected if their influence is considered to be seemingly small.

Model-Based Approach to Organizational Memory

The model-based approach to represent a system can be characterized as trying to build as complete and accurate an internal model of the system as possible and then use this model as the basis for all plans and actions. The elegance and value of internal models has been stated by Craik (1943):

> If the organism carries a small scale model of external reality and of its own possible actions within its head, it is able to try out various alternatives, conclude which is the best of them, react to the future situations before they arise, utilize the knowledge of past events in dealing with the present and the future, and in every way to react in a much fuller, safer, and more competent manner to the emergencies which face it. (p.61).

This statement makes three important assumptions about the properties of a model. (1) Models are small-scale in terms of effort that has to be undertaken to build and maintain their accuracy. (2) Models display predictive capabilities, because the rules and relationship that are the result of past experiences apply also for the future. (3) Models are synthetic. Therefore, their functionality relies basically on how their creator perceives and understands the relationships of the environment the model has to represent. Problems with this model have been identified in their practical application and their validity (Chown, 1999) because the development of a dynamic model is difficult to achieve. Furthermore, the relevance of the inherent knowledge and rules has to be examined constantly in order to maintain the accuracy of the model.

Behavior-Based Approach to Organizational Memory

The difficulty of modeling and representing CASs is caused by the apparently contradictory ambiguity of their characteristics. Because the science of complexity allows the parallel evolvement of contrary theories, common models that represent deterministic processes in an environment with limited variety tend to fail. One logical consequence would be to abandon the usage of models since their predictive capabilities are seemingly small. The alternative is the so-called behavior-based approach (Chown, 1999). It suggests the use of cognitive maps that are rather a product of experience and implicit knowledge than of precise measurement.

In contrast to approaches that mainly rely on a model displaying real world systems, the behavior-based approach assumes that the behavior of an open system emerges from multiple complex interactions. Since CASs employ positive feedback mechanisms, the process of emergence is not reversible. Consequently, it is not possible to isolate and identify a specific cause for a displayed effect. This again means, that a meaningful representation of the system's way of functioning is not derivable from its behavior patterns. If this is assumption applies, organizational memory has to focus rather on storing and representing patterns of behavior rather than patterns of relationships.

As the displayed behavior does not uncover the underlying interactions that have generated the prevailing pattern, organizations will have to maintain parallel interpretations to explain occurring behavior. This may – under specific condition - demand commitments, that are comparable to the effort to maintain the accurate structure of a model-based ontology. Constant monitoring and an evolution-like competition determine whether one or the other interpretation meaningful represents the essence of the environment. However, this approach does not suggests, how this can be achieved by the explicit knowledge base of an organization. The holistic perception of behavior and the potential to interpret surrounding conditions and subsequently identify analogies to past behavior patterns seems to be part of the implicit knowledge. This type of knowledge has been identified as a typical potential of the human brain.

Conclusions

This paper has discussed issues of organizational memory and the challenges that result out of the characteristics of a Complex Adaptive System environment. It has identified the different roles in the human-computer collaboration by defining different types of knowledge. Future developments in Information Technology will move the borderline between these types of knowledge: The share of explicit knowledge within an organization will increase, as sophisticated software applications will enhance their ability to represent complex contents within a computer. The human contribution to organizational memory will translate computer-based knowledge to an organizational context. As the capability to draw analogies and understand patterns is closely linked to the individual skills of individuals, organizations will have to adapt to management practices that are more concerned to fulfill the requirements of these individuals.

Bibliography

Chown, E. (1999, Winter). Making Predictions in an Uncertain World: Environmental Structures and cognitive Maps. Adaptive Behavior, 7 (1), 17-33.

Osterloh, M., and Wübker, S. (1999) Wettbewerbsfähiger durch Prozeß- und Wissensmanagment. Wiesbaden: Dr. Gabler.

Pascale, R. T. (1999, Spring). Surfing the Edge of Chaos. Sloan Management Review, 40 (3), 83-94.

Pohl, J. (2003) The Emerging Knowledge Management Paradigm: Some Organizational and Technical Issues. Preconference Proceedings collaborative Decision-support Systems InterSymp – 2003, Baden-Baden, Germany, 11-26.

Slywotzky, A. J., Mundt, K. A., and Quella, J. A. (1999, June). Pattern Thinking. Management Review, 32-38.

Stacey,R. (1996, May/June). Management and the science of complexity: If organizational life is nonlinear, can business strategies prevail? Research Technology Management, 39, 8-11.

Collaborative Project Delivery

Barry Jones, Ph.D

Construction Management Department, College of Architecture and Environmental Design, California State Polytechnic University, San Luis Obispo, California

Introduction

The construction industry, as we see it now in the 21st Century, is still characterized by a fixed price project delivery system that often brings an adversarial culture to the project rather than maximizing client value.
However, action on the recommendations from key Construction Industry strategic reports has begun to have some effect on changing the practice of project delivery. The reports include the U.S. Construction Industry Institute (CII) report ''In Search of Partnering Excellence'' (CII 1991), and the initiatives for change in construction set out in the ''Joint Review of Procurement and Contractual Arrangements in the United Kingdom Construction Industry'' (Latham 1994) and the Construction Task Force report (Egan, 1998).
The Chartered Institute of Building that represents the construction industry in the UK and has a membership worldwide is also driving change through their "Accelerating Change" strategy[1]. A recent press release from the CIOB dated April 2004 indicates the commitment of the UK Government to this process *"Construction Act set for Sir Michael Latham review Construction Minister, Nigel Griffiths has appointed Sir Michael Latham to embark on a review into Part II of the Housing Grants Construction and Regeneration Act 1996……. The provisions in the Construction Act provide the basis for a fairer payment culture in construction and more effective project delivery. I want to make sure we continue to bring about improvements in practices through all means available and with the support of all sectors of the industry. Sir Michael's appointment will help make this happen."*

With larger multi-national clients of construction services the call for greater value is one echoed around the world. For instance the construction industry in Singapore is similarly faced with a multitude of problems which have affected its performance. To address these multi-faceted problems, a major review of the construction industry was initiated in May 1998. This review, which involved captains from all segments of the construction industry both in the public and private sectors, resulted in the release of the Construction 21 (C21) Report - a blueprint to chart the future directions of Singapore's construction

[1] In the last 3 years the M4I demonstration projects: Represent 3% of industry output; Have killed ZERO people rather than 8; 30% better on Time and Cost predictability; 32% better on Quality; 29% higher productivity; 33% higher Client satisfaction; 35% higher profitability. Ref: Alan Crane, Chair, *Rethinking Construction*
In the last 3 years the M4I demonstration projects:77% report they have learned or improved partnering skills, supply chain management or procurement processes; 60% report improved ability to plan projects and achieve the plans.
DTI PII funded research led by a leading contractor

sector. Recommendations of the Report are now being implemented by both the public and private sectors. Among other things, the C21 Report identified "An Integrated Approach to Construction" as one of the several strategic thrusts that the industry needs to embrace to achieve greater synergy, higher productivity and quality.

The core concern of these reports was the ability of the International construction community to deliver a high quality product to its clients in the 21st Century. The *Latham Report* was seen by many as a turning point for the construction industry, radically transforming relationships between clients and contractors.

Targets Set for the Construction Industry

The Report recommended that contracts should be founded upon principles of fairness, mutual trust and teamwork with greater synergy of complementary roles of the different participants. The improvements targets (Table 1) set are:

Table 1 - Construction Sector Performance Improvement Targets set to be achieved by 2000 - USA and UK

Construction Sector Performance	USA		UK
	Target	Rank	Target
Total Project Delivery Time	Reduce by 50%	First	Reduce by 25% over 5 years 10%/year
Lifetime Cost (Operations Maintenance Energy)	Reduce by 50%	Second	
Productivity and Comfort Levels of Occupants	Increase by 50%	Fifth =	Improve by 20%
Occupants Health and Safety Costs	Reduce by 50%	Sixth	
Waste and Pollution Costs	Reduce by 50%	Fifth =	
Durability and Flexibility in Use over Lifetime	Increase by 50%	Third	
Construction Worker Health and Safety Cost	Reduce by 50%	Fourth	
Costs			Reduce by 30% over 5 years 10%/year
Construction Quality			Zero Defects
Building defects			Reduction 20%/year

The source information: USA – Wright Rosenfield Fowell, 1995; UK - The Engineering and Physical Science Research Council's Innovative Manufacturing Initiative Programmed.

With these recommendations and targets in mind, and projects evermore complex, uncertain, and pressed for speed, there is a need for a project-based production system to maximize value and minimize waste. The challenge requires a restructuring of the project delivery process to find better ways in which all the key participants can work together, with the client's interests being central to the process. The new system must allow early consideration of issues such as buildability, construction quality and safety, environmental performance, maintainability, life cycle costing, use of IT for project integration etc. With traditional delivery systems, at the design stage, inputs from contractors and suppliers are seldom sought, very often leading to frustration, subsequent re-work and delays in project execution when the expectations of the designers and contractors do not meet. At the extreme, clients, consultants and contractors may end up in confrontation or litigation.

Changing the Mindset

Architects, engineers and construction managers are trained and work in different ways. Attraction to their individual professions and career paths can often be traced back to their different personality types. The architect as the social artist, the engineer as the mathematician proving structures will stay upright and the construction manager as the team-minded resource co-coordinator, the people person. Over many decades these differences in mindset have caused the project process to falter rather than advance for the common good of the project and client. All must change their mindset to create a positive constructive project innovative environment.

You can recognize a company with an innovation mindset by the way employees interfaces with each other. They treat each other with respect, admiration, and cooperation. They smile. They laugh. They express consideration and thoughtfulness to each other. They listen. They focus on the benefits desired by consumers rather than on their own personal gain. They come to work with an optimistic enthusiasm because they believe that what they do each day really does count. They focus on the future rather than on the past. They exude self-confidence, possess a healthy self-esteem, and believe in their own capabilities and strengths. They have faith in innovation and in each other.

An innovation mindset is an attitude that should be adopted throughout an organization by virtually every employee, from the CEO to hourly workers. While a mindset has to exist in individuals, it can spread and be adopted and nurtured by others. It is a pervasive aura which has a spirit of its own. This mindset stimulates and motivates individual employees, as well as teams, to holistically endorse a belief in creating newness.

Identifying innovation values and new product team norms to guide behavior and communications among team members is crucial. Determine individual team member goals, hopes, fears, and aspirations are essential. You need to have each individual member discuss with the entire team his or her reasons for participating in the development of new project. Each one of them needs to articulate what he or she wants to get out of it -- personally. Companies that allow teams to invest adequate time up front to

do this, and are open to the inputs made, help to solidify and empower new product/project teams.

Partnering - Design Build

Advocates know the primary benefits of teamwork versus an adversarial relationship, which enable decisions to be made in a much timelier manner, and of course, dispute avoidance, for the sake of savings of time and money during construction. Over the past decade two delivery strategies that have been widely adopted by construction clients to procure their new buildings and structures include partnering and design/build.

The partnering philosophy has been a major concept in the worldwide effort of creating significant improvements to the construction industry and changing the mindsets of participants to the project process. Since the U.S. Construction Industry Institute (CII) report "In Search of Partnering Excellence" (CII 1991), and the initiative for change in construction set out in the "Joint Review of Procurement and Contractual Arrangements in the United Kingdom Construction Industry" (Latham, 1994) and the Construction Task Force report (Egan 1998), partnering has begun to have some positive effect.

The US Army Corps of Engineers through 'The Project Partnering Process' creates a new team building environment which fosters better communication and problem solving, and a mutual trust between the participants. These key elements create a climate in which issues can be raised, openly discussed, and jointly settled, without getting into an adversarial relationship. Through this process of teamwork and problem solving on a construction project, the Corps goals are in the areas of Safety, Quality, Schedule, Budget, and Disputes. They want the quality of the work to be right the first time, the project to be completed on time, the final cost to be within budget, and disputes/litigation to be minimized. The goals of the contractor are very similar, thus the process benefits both parties through the teamwork and pursuit of mutual goals. The use of formal and informal partnering techniques now has widespread use across the Corps during the construction phase of our projects, and has been adopted by many Federal, states, and local agencies based on the Corps success. They see partnering at the project level, or at any level for that matter, brings a synergy to the project delivery team, which is unmatched in effectiveness and benefits to projects, and ultimately to customers/users of constructed facilities.

The C21 Report has identified Design-and-Build (D&B) as a form of procurement which can play a positive role in encouraging integration among the project team members. Compared with the traditional Design-Bid-Build procurement system, D&B will foster the integration of the expertise of the consultants and contractors at an early stage to incorporate build able design and more innovative construction methods to save cost and labor, minimize wastage etc.

In Australia and Japan, D&B projects account for about 60% and 50% of projects respectively, in the UK around 40%. However, in the USA the figure is much smaller at about 20% and in Singapore, the D&B method of procurement is still not the preferred

choice especially amongst the private sector clients and accounts for only about 14% of total projects. If more integration in the project delivery processes is the way to go, there is then a need to look at ways and means to promote the D&B method and to eliminate practices which inhibit the adoption of such a method.

Building a Collaborative Problem Solving Environment

With the above drive for change in project delivery it led the author's research direction to find a way in which all participants to the construction project can input their domain knowledge to solve collaboratively design and production problems at all stages of the project supply chain.

The partnership environment proposed is one that fully utilizes the strengths of a multi-agent computer environment collaborating with the various human domain experts. During the life of the project there will exist a total problem solving environment where the knowledge and intelligence of all domain-contributing agents can be fully employed. Better opportunities therefore exist to concurrently view the effect of decisions that impinge on the many domain participants. All contributors are collaboratively drawn into the problem solving process. Time is saved because a concurrent collaborative problem solving approach is adopted rather than the traditional sequential problem solving approach.

Experts can still be geographically or functionally distributed; this also presents the opportunities to take advantage of recent technology in communication systems (co-operative distributed, broad band, etc.). The complexities of design and production can be broken down over numerous agents; problems can be decomposed to a level at which computer agents can contribute essential knowledge. Systems architecture will be designed to link relational databases of essential domain knowledge. The environment proposed could be extended to continually monitor and assist throughout the life cycle of construction projects.

Figure 1 (appendix 1) outlines to problem solving environment proposed. Figure 2 (Appendix 2) portrays how the 'family' of Construction Management agents might devolve the problems in partnership with families of Architectural, Engineering, Planning, Client, Electrical and many other domain representative agents all collaborating to solve the problems effecting the project.

The Problem Solving Environment in Action

Within the computer agent environment, problem solving is seen as a co-operative process with mutual sharing of information to produce a solution. Solutions result from an assembly of construction objects, e.g. bricks, walls, floors, windows, etc., these are assembled by human and computer agents to satisfy project specific criteria, e.g. quality, environmental, cost, safety, etc. Objects are information entities only whereas computer agents are active and have knowledge of their own nature, needs and global goals. Objects are accessible by agents but cannot take action. Having this ability to view the

artifacts used in the project model as a series of objects, which have implicit attributes and features, gives scope to analyze the design with regard to such aspects as manufacture, constructability, cost, quality, safety, etc., an almost unlimited definition of machine agents could be specified that are caretakers of knowledge pertaining to most of the constraints and criteria related to a new building project.

In such an environment the design facilitator's role would be one of searching, evaluating and modifying the current design and production state with the support of different domains computer agent families (Jones, 1994). In this process the various expert human agents would direct and guide the efforts of all computer agents to advance the current state towards a best solution that is acceptable to all domains agents. The role of the human co-coordinator would be that of principal long term or strategic planner while agents would focus mainly on short-term activities, and therefore should be endowed with knowledge that enables them to only execute short term and reactive plans. The characteristics such computer agents would possess are:

(a) Programmed with appropriate problem solving protocols.

(b) Intelligience[2] in that they possess the capacity to plan their own actions. Intelligent agents would therefore have implicit domain knowledge, knowledge of their own needs, knowledge of global goals, the ability to communicate and the ability to take action. They would also have access to objects, which are information entities, but unlike agents, cannot take action.

(c) Belong to domain families, each family of computer agents and objects would represent each domain and their problem solving activities associated with the design and production problem solving of that specific project. As other problems arise so the agent environment would extend or should the project be of a different construction then a new agent family would be appropriately designed.

(d) The ability to decompose the problem to a level it can be solved including recognition of the requirement to seek collaboration from other different domain agent families that collectively are required to solve the problem.

(e) Operate in a narrow domain providing support to requests for assistance. Agents would range from simple to complex processing units each rationally working toward a single global goal or towards separate individual goals that interact. Acting independently in a self-regulating manner their common purpose is to change the current design state towards meeting a common set of

[2] Intelligence in the context of this work implies that the design system has some means that allows it to anticipate the data needs, information needs or knowledge needs of the human designer. The system would act as an intelligent assistant to the evolving design, aiding the designer and freeing them from being overwhelmed with untimely knowledge. Providing such assistance to all problem solvers in the design environment requires an understanding of the various participants' knowledge, factors that constrain their decisions and criteria they work under. Pohl (1993) called this an Intelligent Computer Assisted Design System (ICADS). The ICADS approach is supported in several working models (ICADS-DEMO1 (Pohl,1989), ICADS-DEM0 2 (Pohl, 1991), AEDOT (Pohl, 1992). These have provided computer scientists with a useful test bed for the development of a body of knowledge relating to software and hardware computer architecture, theoretical concepts and technical implementation issues.

goals. The goals are set by the human agents with advice from various autonomous agents that include agent representation of the client.

(f) Agents would use their local expertise and available resources to work in parallel on different or co-coordinating tasks to arrive at a solution in the following ways:

(i) Act as co-operative search agents that liaise with knowledge bases in the search for alternative solutions.

(ii) Act as evaluators and solution proposes to express opinions about the current state of the design solution.

(iii) Give continuous background monitoring and evaluation of the evolving design solution.

(iv) Designed to have implicit domain knowledge, knowledge of their own needs, and knowledge of global goals, the ability to communicate and the ability to take action.

(v) Each agent would be represented at the level of detail at which the design facilitator or human agent wishes to reason about the project problem solving system.

A coordinator should be capable of invoking a procedure for resolving conflict conditions based on consultation. The agents use their specialized expertise and available resources to work in parallel on different or coordinating tasks to arrive at a solution concurrently. There is an inevitable need for interaction between all the participants who input to complete the final project. Pohl (2000) states that the computer system should reflect the more realistic situation of a project team, one that interacts by co-operation and persuasion. Complete families of computer-agents that represent a particular domain can be built e.g. architect, interior designer, structural engineer, landscape architect, safety manager, quality manager, environmental manager, mechanical and electrical engineer, construction manager, project manager, etc. and within each family specific agents would monitor and offer assistance regarding criteria and constraints imposed in the areas of environmental, quality, safety, cost, production time, etc. There could be a 'Quality' agent residing in a number of domains i.e. Architect, Construction manager, Project Manager, Quality manager, each would be representing the criteria and constraints related to quality of that domain.

Conclusion

The author has portrayed and conceptualized a collaborative problem solving system that will facilitate a project delivery based on integration between all the key participants to the project. Appendix 3 indicates the future in project delivery that gives the client value as proposed by the various reports and strategies touched on in this paper.

References

Audit Commission. (1997). *Rome wasn't built in a da*y, Belmont Press, Northampton.

Brooke, K. L., and Litwin, G. H. (1997). ''Mobilizing the partnering process.'' *J. Mgmt. in Engrg*., ASCE, 13(4), 42–48.

Construction Industry Board. (1997). *Constructing succes*s, Thomas Telford, London.

Construction Industry Council (CIC). (2000). ''A Guide to Project Team Partnering.'' London.

Construction Industry Institute (CII). (1991). ''In search of partnering excellence.'' *Spec. Publ. 17-1 Prepared for Constr. Industry Inst*.,Partnering Task Force.

Construction 21 (C21) Report - a blueprint to chart the future directions of Singapore's construction

Egan, J. (1998). ''The report of the Construction Task Force: Rethinking construction.'' Department of the Environment, Transport and the Regions, London.

Jones. B.K (1998) "A Model for Collaborative Engineering in the Construction Industry", Thesis for Doctor of Philosophy, Dept. of Civil and Environmental Engineering, University of Southampton, Southampton, UK.

Jones.B.K. and M.J.Riley (1995) Collaborative Construction Agents, ASCE, 2nd International Congress on Computing in CE, Atlanta, USA, June 1995, pp 1316-1323. International Conference, Technical paper and presentation.

Jones.B.K. and M.J.Riley (1995) Autonomous Construction Agents in an ICADS environment, ASCE,6th International Conference in Civil and Building Engineering, Berlin, Germany, July 1995, pp 407-412. Technical paper and presentation.

Jones, B.K. and M.J. Riley (1994) Construction Problem Solving in a Co-operative Distributed Agent Centered Environment, 1st congress on Computing in C.E. Washington D.C.

Larson, E. (1997). ''Partnering on construction projects: A study of the relationship between partnering activities and project success.'' *IEEE Trans. on Engrg. Mgmt*., Piscataway, N.J., 44(2), 188–195.

Latham, M. (1994). *Constructing the tea*m, Her Majesty's Stationery Office, London.

Pohl, J., L. Myers, A. Chapman, and J. Cotton (1989). "ICADS: Working Model Version 1," Technical Report, C,ADRU-03-89, CAD Research Unit, Design Institute, School of Architecture and Environmental Design, Cal Poly, San Luis Obispo, California.

Pohl, J., L. Myers, A. Chapman, J. Snyder, H. Chauvet, J. Cotton, C. Johnson and D. Johnson (1991). "ICADS Working Model Version 2 and Future Directions," Technical Report, CADRU-05-91, CAD Research Center, Design Institute, College of Architecture and Environmental Design, Cal Poly, San Luis Obispo, California.

Pohl, J., L. Myers, J. Cotton, A. Chapman, J. Snyder, H. Chauvet, K. Pohl and J. La Porta (1992). "A Computer-Based Design Environment: Implemented and Planned Extensions of the ICADS Model," Technical Report, CADRU-06-92, CAD Research Center, Design and Construction Institute, College of Architecture and Environmental Design, Cal Poly, San Luis Obispo, California.

Pohl,J and L. Myers (1993) A Distributed Cooperative Model for Architectural Design, CAD Research Centre, Cal. Poly, San Luis Obispo, CA.

Pohl,J., A Chapman,. K Pohl (2000) Computer-Aided Design Systems for the 21st Century: Some Design Guidelines, Collaborative Agent Design (CAD) Research Center, San Luis Obispo, CA.

Pohl (2001) Information-Centric Decision-Support Systems: A Blueprint for 'Interoperability', Office of Naval Research: Workshop on Collaborative Decision-Support Systems, Quantico, VA, June 5-7, 2001

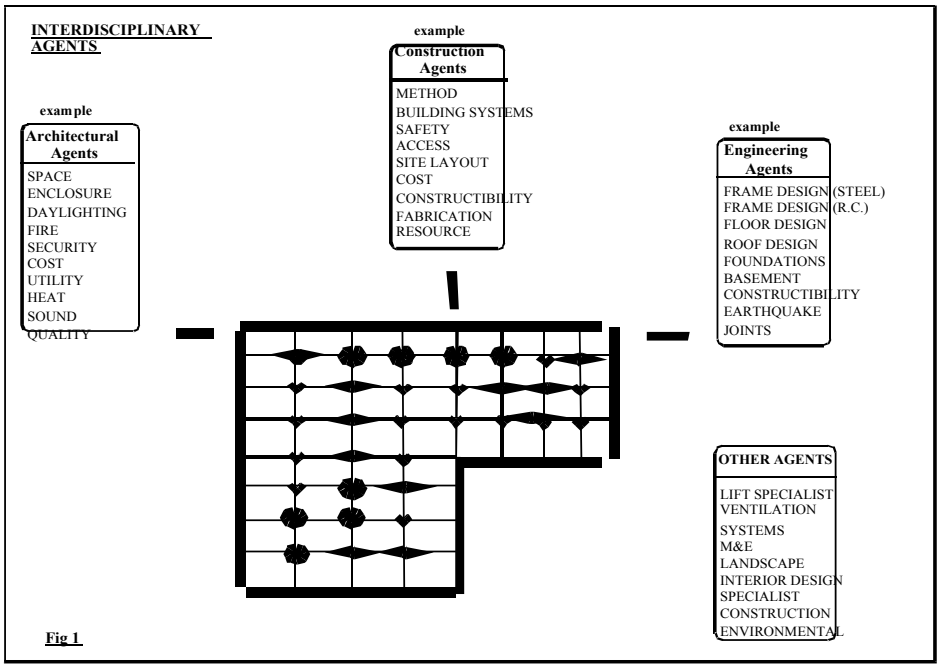
Puddicombe, M. S. (1997). ''Designers and contractors: Impediments to integration.'' *J. Constr. Engrg. and Mgmt*., ASCE, 123(3), 245–252.

Towill, D. R. (1997). ''Successful business systems engineering.'' *IEE Engrg. Mgmt. J*., 7(1), 55–64.

Wright, R. N., Rosenfield, A. H., and Fowell, A. J. (1995). ''National Science and Technology Council report on federal research in support of the U.S. construction industry.'' Washington, D.C.

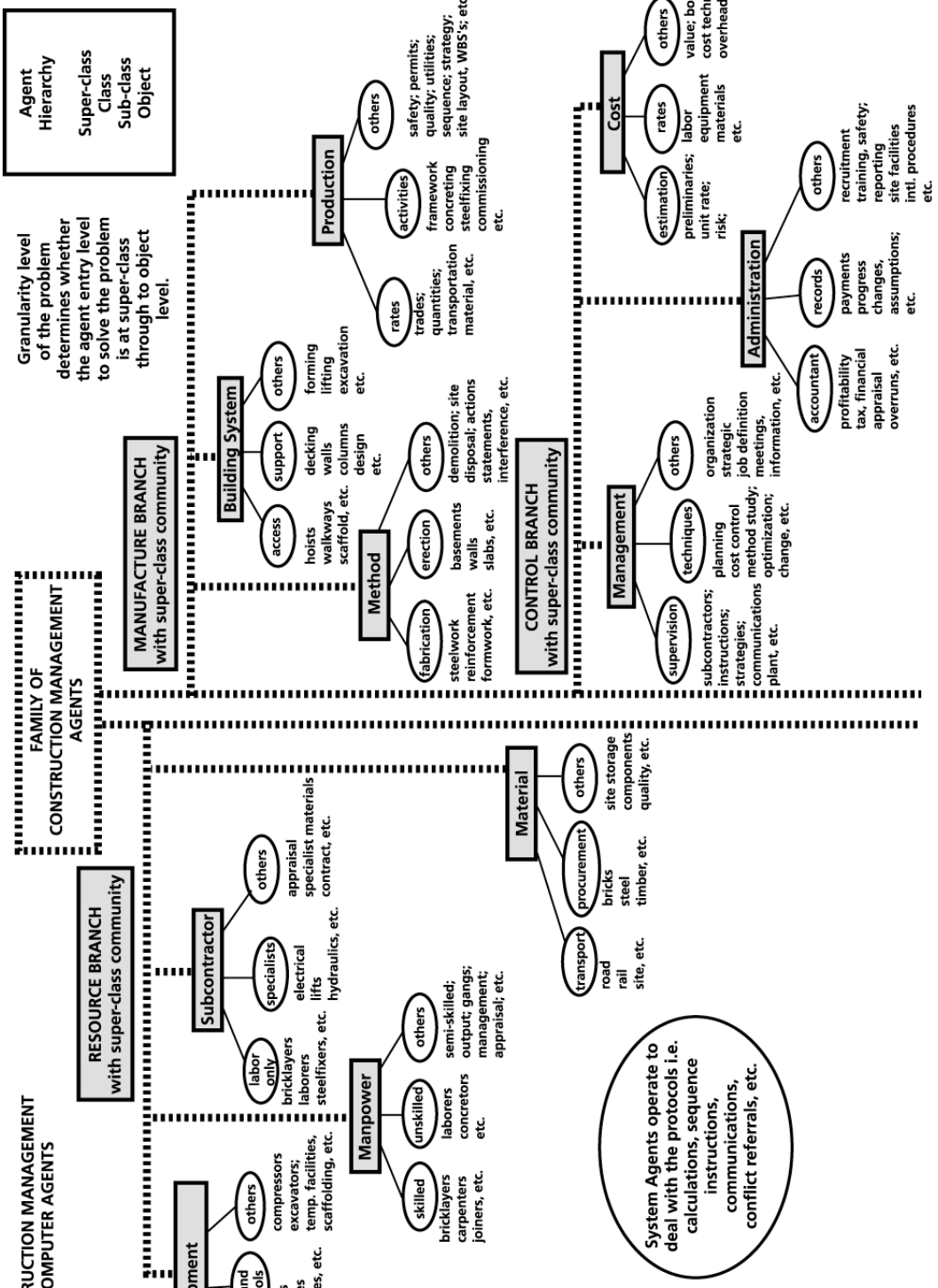
Appendix 1

Figure 1 – Interdisciplinary Collaborating Agents

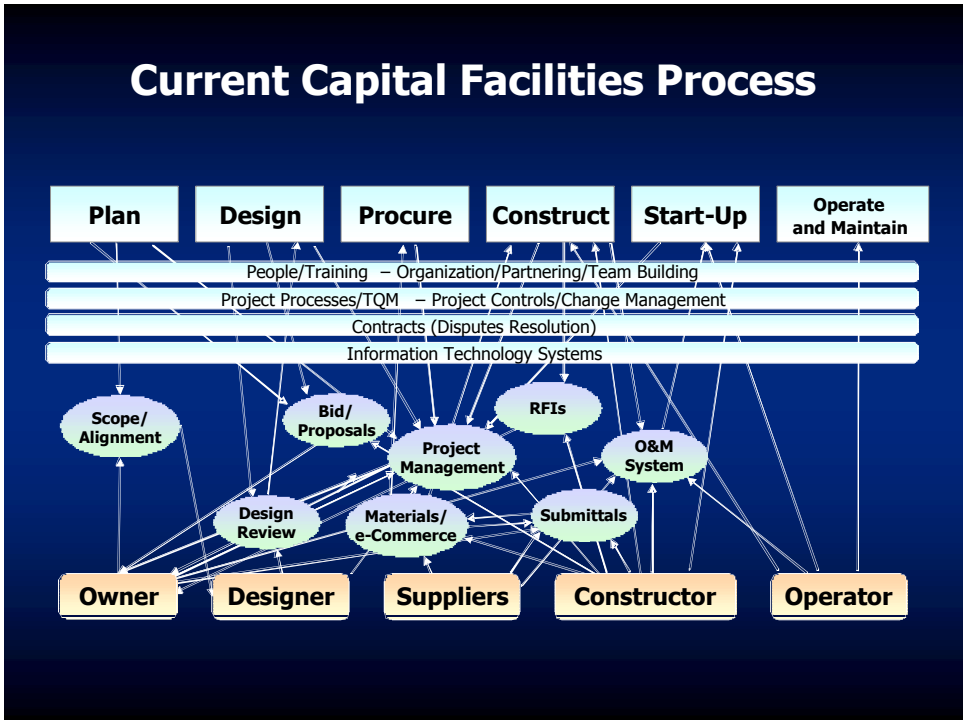


Appendix 2

Figure 2 Computer Agent Families - Construction Management



Appendix 3


Current Capital Facilities Process
Plan
Design
Procure
Construct
Start-Up
Operate and Maintain
People/Training – Organization/Partnering/Team Building
Project Processes/TQM – Project Controls/Change Management
Contracts (Disputes Resolution)
Information Technology Systems
Scope/ Alignment
Bid/ Proposals
RFIs
Project Management
O&M System
Design Review
Materials/ e-Commerce
Submittals
Owner
Designer
Suppliers
Constructor
Operator