Challenging Computer Software Frontiers and the Human Resistance to Change

Jens Pohl, Ph.D.

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

Abstract

This paper examines the driving and opposing forces that are governing the current paradigm shift from a data-processing information technology environment without software intelligence to an information-centric environment in which data changes are automatically interpreted within the context of the application domain. The driving forces are related to the large quantity of data and the complexity of networked systems that both call for software intelligence. The opposing forces are non-technical and due to the natural human resistance to change.

Based on this background the paper describes current information-centric technology, proposes a vision of intelligent software system capabilities, and identifies four areas of necessary research. Most urgent among these are the ability to dynamically extend and merge ontologies and semantic search capabilities that can be initiated either by human users or software agents. Longer term research interests that pose a more severe challenge are related to the translation of emerging theoretical *hierarchical temporal memory (HTM)* concepts into usable software capabilities and the automated interpretation of graphical images such as those recorded by surveillance video cameras.

Keywords: agents, cognition, context, data-centric, extensible ontologies, human nature, HTM, image interpretation, information-centric, paradigm shift, representation, resistance to change, semantic search, situatedness, software, SOA, TEGRID.

Periods of accelerated change

Over the past hundred years there have been many fundamental changes in our human values and the way we perceive our environment (Figure 1). The Industrial Age placed great value on physical products and devised ingenious ways to maximize the manual contributions of its human work force in a subservient role to a highly automated mass production process. In the Information Age the focus has moved from the physical capabilities of the human work force to the intellectual capabilities and potential of its individual members. The attendant symptoms of this profound shift are the replacement of mass production with computer controlled mass customization, virtual products as opposed to physical products, and the creation and exploitation of knowledge. However, the rate of change is by no means constant.

Throughout history there have been periods of rapid and profound change. More often than not, and certainly in recent times, the precipitating factors have been technological and/or political in nature. Sometimes these factors have gained momentum over time in a cumulative manner such as the French Revolution in the 18th Century, and at other times they have descended on society

more abruptly. The terrorist attacks on the United States (US) that occurred on September 11, 2001 (9/11) are an example of the latter. In either case such periods of change have typically been accompanied by a great deal of human tension.

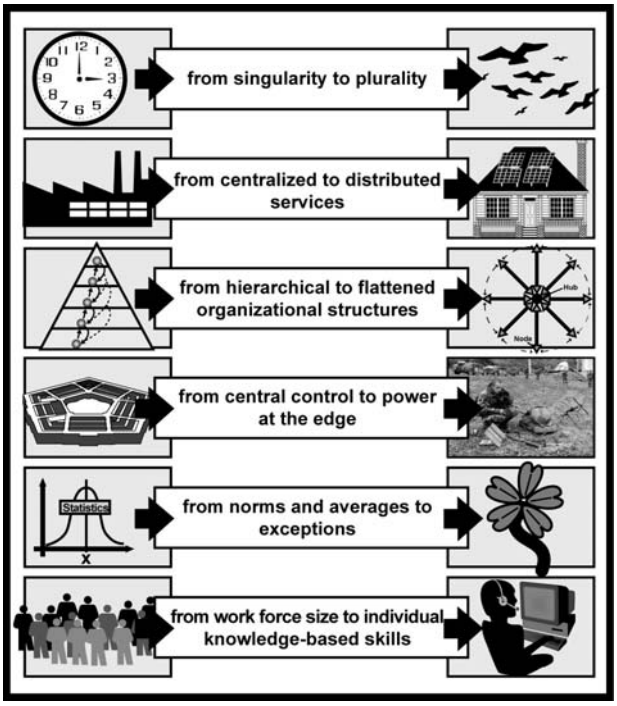

Figure 1: Many fundamental changes

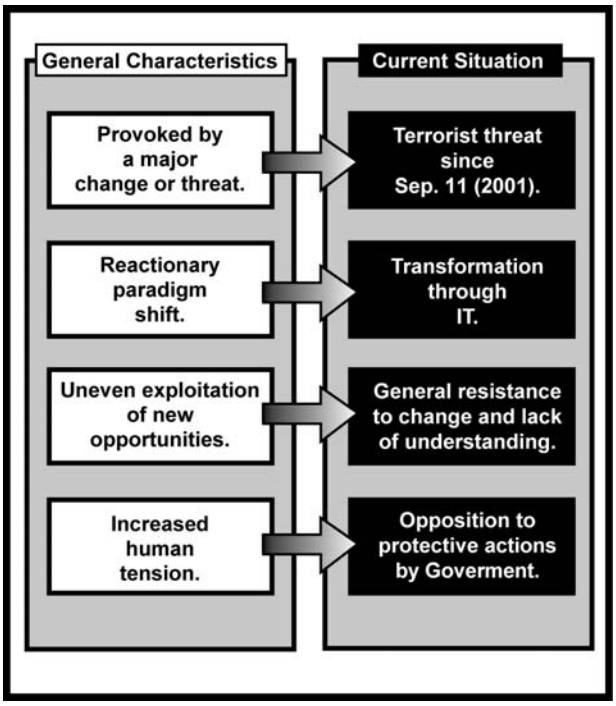

Figure 2: Periods of accelerated change

It is the dual purpose of this paper to explore some of the underlying reasons for the tensions that accompany periods of rapid change and to discuss the technological advances in computer software that are emerging as a natural byproduct. These advances tend to fall into two categories, namely: the implementation of theories and methodologies that have been under development for some time but were not exploited because there did not appear to be a compelling need for their immediate application; and, requirements for additional advances that become apparent as this existing knowledge transitions from focused research projects to broader and larger scale utilization. Typically, the first category manifests itself as a paradigm shift that is accompanied by an order of magnitude increase in capabilities and inevitably demands fundamental changes in the performance and management of existing tasks. The second category becomes apparent as human expectations for higher levels of exploitation of the new capabilities identify the need for additional capabilities.

The origin of a paradigm shift is normally associated with compelling needs that are often of a threatening nature (Figure 2). To counter such threats society is forced to be critical of existing methodologies and processes, to be innovative, and to seek new capabilities that will improve its chances of survival. Therefore, the paradigm shift itself is borne out of fear as the primary source of tension. In the post-9/11 world the US Government found it necessary to initiate a degree of mobilization and reorganization that was unprecedented since World War II. In particular, the urgent requirement to protect the public from terrorist threats focused attention on information systems for identification, surveillance, and intelligence gathering purposes. It was soon realized that due to the enormous quantity of data involved the computer-based information systems

would need to be able to assist the human users in the interpretation of the data that they are processing. This requirement has initiated a paradigm shift from computer-based data-processing to intelligent information management.

A secondary source of tension soon arises as further technical challenges and opportunities for the increased exploitation of the new capabilities emerge. This source of tension is not as severe as the primary forces that precipitated the paradigm shift because it is more narrowly focused on the research community and its funding organizations. The additional capabilities that become available tend to be incremental in nature and are therefore perceived to be less disruptive. Even though these complementing innovations may be even more profound in their enabling capabilities, since society is already engaged in a paradigm shift they become part of the mainstream of change and are therefore more readily accepted. In the post-9/11 world these emerging research challenges are related to the development of software methodologies that will improve the versatility and reliability of the automated transformation of data into actionable information and the intelligent management of this information.

Humans are *situated* in their environment

To explore the source of the resistance to change and attendant tensions that inevitably accompany a paradigm shift it is necessary to understand that we human beings are very much influenced by our surroundings. As shown in Figure 3, we are *situated* in our environment not only in terms of our physical existence but also in terms of our psychological needs and understanding of ourselves (Brooks 1990). We depend on our surroundings for both our mental and physical wellbeing and stability. Consequently, we view with a great deal of anxiety and discomfort anything that threatens to separate us from our environment, or comes between us and our familiar surroundings.

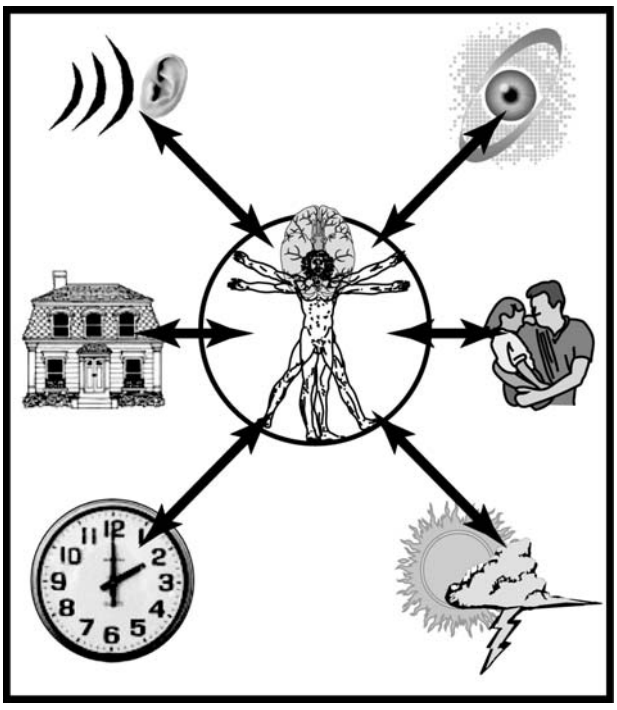

Figure 3: Situated in our environment

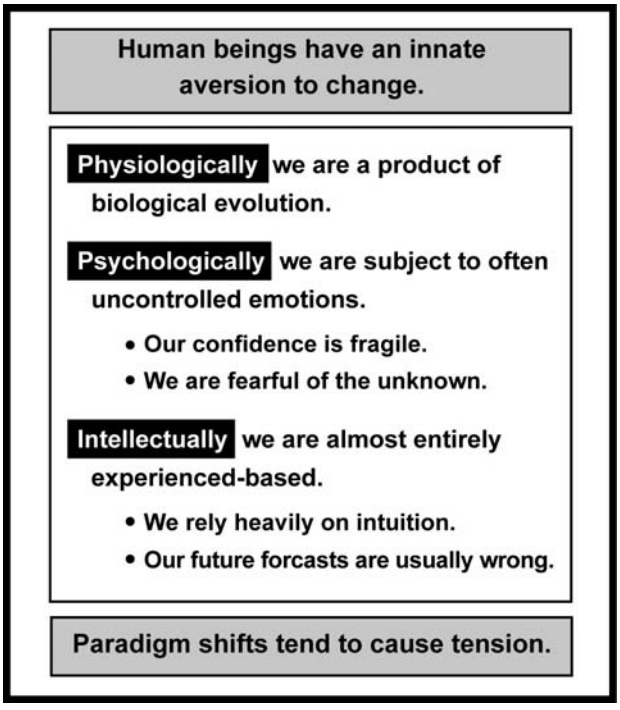

Figure 4: Human resistance to change

This extreme form of *situatedness* is a direct outcome of the evolutionary core of our existence. The notion of evolution presupposes an incremental development process within an environment that represents both the stimulation for evolution and the context within which that evolution takes place. It follows, first, that the stimulation must always precede the incremental evolution that invariably follows. In this respect we human beings are naturally reactive, rather than proactive. Second, while we voluntarily and involuntarily continuously adapt to our environment, through this evolutionary adaptation process we also influence and therefore change our environment. Third, our evolution is a rather slow process. We would certainly expect this to be the case in a biological sense. The agents of evolution such as mutation, imitation, exploration, and credit assignment, must work through countless steps of trial and error and depend on a multitude of events to achieve even the smallest biological change (Waldrop 1992, Kauffman 1992, Holland 1995, Pohl 1999).

In comparison to biological evolution our brain and cognitive system appears to be capable of adapting to change at a somewhat faster rate. Whereas biological evolution proceeds over time periods measured in millenniums, the evolution of our perception and understanding of the environment in which we exist tends to extend over generational time periods. However, while our cognitive evolution is of orders faster than our biological evolution it is still quite slow in comparison with the actual rate of change that can occur in our environment.

Human resistance to change

Clearly, at least in the short term, the experience-based nature of our cognitive system creates a general resistance to change (Figure 4). The latter is exacerbated by a very strong survival instinct that manifests itself in a desire for certainty as a source of absolute security (Figure 5). Driven by the desire to survive at all costs we hang onto our past experience as insurance. In this respect much of the confidence that we have in being able to meet the challenges of the future rests on our performance in having met the challenges of the past (i.e., our success in solving past problems). We therefore tend to cling to the false belief that the methods we have used successfully in the past will be successful in the future, even though the conditions may have changed. As a corollary, from an emotional viewpoint we are inclined to perceive (at least subconsciously) any venture into new and unknown territory as a potential devaluation of our existing (i.e., past) experience.

This absolute faith in and adherence to our experience manifests itself in several human behavioral characteristics that could be termed limitations. First among these limitations is a strong aversion to change. Typically, we change only subject to evidence that failure to change will threaten our current existence in a significant way. The current paradigm shift from data-centric to information-centric computer software serves as an example. Although the digital computer was originally conceived as a very fast computational machine capable of reducing the time required for the solution of large numbers of mathematical equations from days to seconds, it soon emerged as a data storage and processing facility. This was mainly due to the need for record keeping accelerated by the growth of commerce and industry driven by major improvements in the ability to travel and communicate over long distances. As a result new opportunities for interaction, leading to cooperation, and eventually collaboration, presented themselves. As the intensity of these activities and the tempo of daily life increased so also did the competition among the human players. However, it did not occur to these players for at least

two decades that the functions of the computer could extend beyond the rote storage and processing of data to the representation of information as a basis for automatic reasoning capabilities.

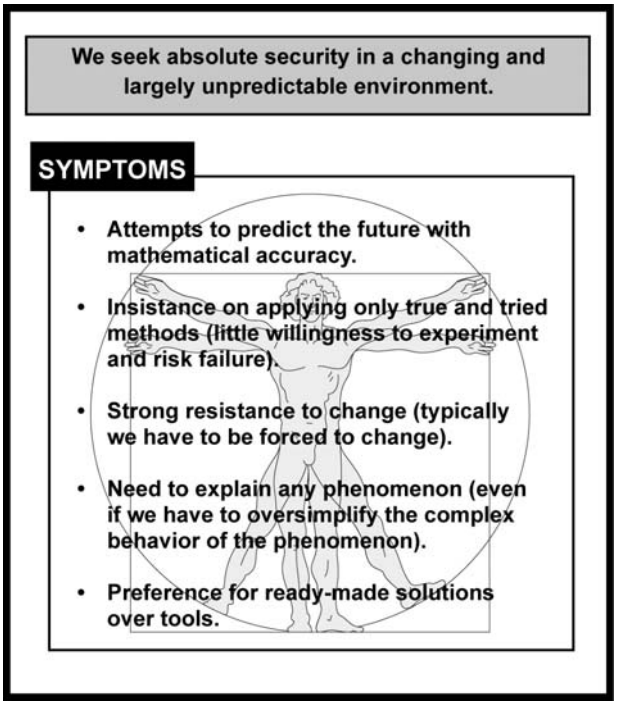

Figure 5: Insecurity as a source of tension

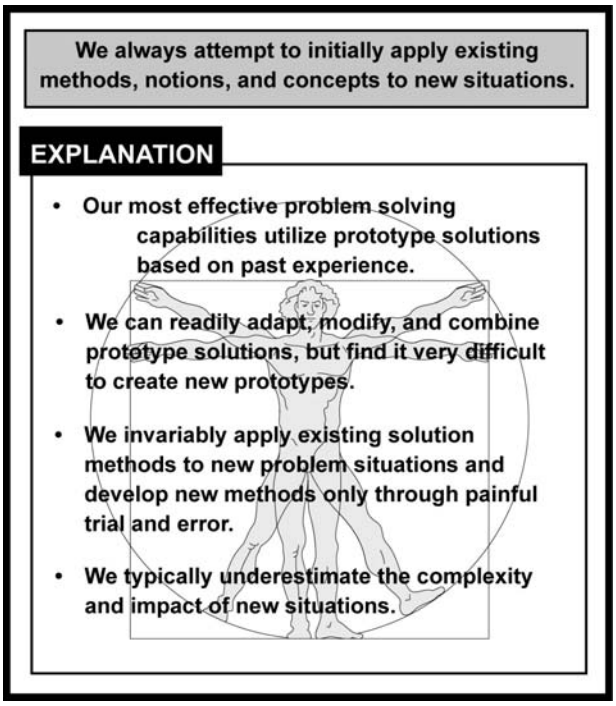

Figure 6: Dealing with new situations

Prior to the events of 9/11 the gradual realization that human-computer interaction could be raised to the level of meaningful collaboration came not as a result of creative discovery, but because the requirement of interpreting the vast amount of computer-stored data simply outstripped the availability of human resources. In other words, it was not the opportunity for using computers in this far more useful role, but the necessity of dealing with an overwhelming volume of data that was gradually persuading computer users to elevate data-processing to information representation in support of automatic reasoning capabilities. Subsequent to 9/11 the absolute necessity of automating at least the lower levels of intelligence gathering and analysis has begun to accelerate the transition from persuasion to conviction. Driven by the realization that the US can no longer afford to depend on the mostly manual processing of intelligence data, key government officials responsible for implementing a vastly improved infostructure have begun to seriously pursue an information-centric software architecture (Cooper 2002).

A second limitation is our apparent inability to resist the temptation of applying old and tried methods to new situations, even though the characteristics of the new situation are actually quite unlike the situations in which the existing methods were found to be useful (Figure 6). This typically casts us into an involuntary experimental role, in which we learn from our initial failures. Examples abound, ranging from the development of new materials (e.g., the flawed introduction of plastics as a structural building material in the 1950s) to the reluctance of the military to change their intelligence gathering and war fighting strategies long after the conclusion of the Cold War era in the 1990s (Wood 2001).

A third limitation is our tendency to view new incremental solutions as final comprehensive solutions. A well known example of such a problem situation was the insistence of astronomers from the 2nd to the 15th Century, despite mounting evidence to the contrary, that the heavenly bodies revolve in perfect circular paths around the Earth (Taylor 1949, 108-129). This forced astronomers to progressively modify an increasingly complex geometric model of concentric circles revolving at different speeds and on different axes to reproduce the apparently erratic movement of the planets when viewed from Earth. Neither the current scientific paradigm nor the religious dogma of the church allowed the increasingly flawed conceptual solution of Ptolemaic epicycles to be discarded. Despite the obviously extreme nature of this historical example, it is worthy of mention because it clearly demonstrates how vulnerable the rational side of the human cognitive system is to emotional influences (Pohl et al.1997, 10-11).

The current paradigm shift

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. Advances in computer hardware 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 (Figure 7).

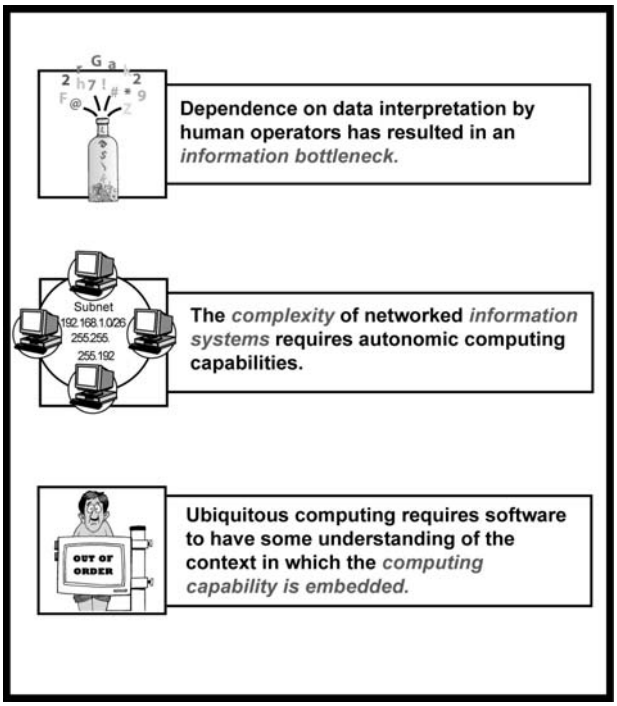

Figure 7: Why do we need context?

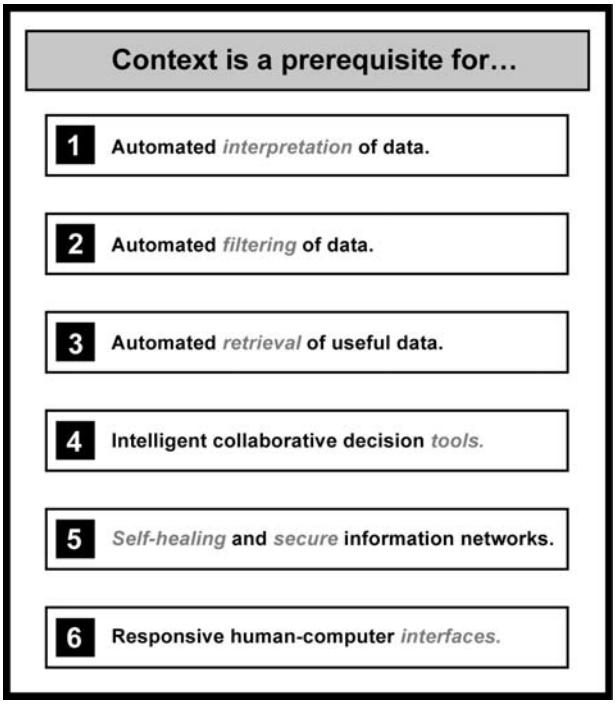

Figure 8: Where should we apply context?

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.

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, when the real world conditions vary from those design specifications. Instead, what is needed is software that incorporates tools that can autonomously adapt to changes in the application environment (Figure 8).

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. Several decades before the sobering events of 9/11 the theoretical foundations were laid for the transition from data-processing to information-centric computer software. 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.

Computer software research challenges

An information-centric computer-based environment extends beyond the ability to automatically interpret data into areas that are related to interoperability, flexibility, intelligent analysis and evaluation capabilities, discovery, and security. Combined with the principles of a service-oriented architecture (SOA) in a distributed implementation, the vision that emerges is profoundly different from the vast majority of existing software systems.

What is suggested is a software environment in which functional capabilities are seamlessly available without the user being aware whether a particular capability is provided by one or more services that are internal to the enabling environment or by an external legacy application that is being accessed through an interoperability bridge. Any data that are being exchanged among internal or external services are shared within the context from which the data derive meaning. The services themselves are not necessarily preconfigured but may be discovered during execution on an as-needed basis. This implies that services are able to automatically configure themselves in conformance with the operational environment and the governing interface protocols.

All of these capabilities are essentially technically feasible today and form part of the notion of a SOA. This notion is by no means new in the software industry, however, it was not until web services came along that SOA principles could be readily implemented (Erl 2005). Initial

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

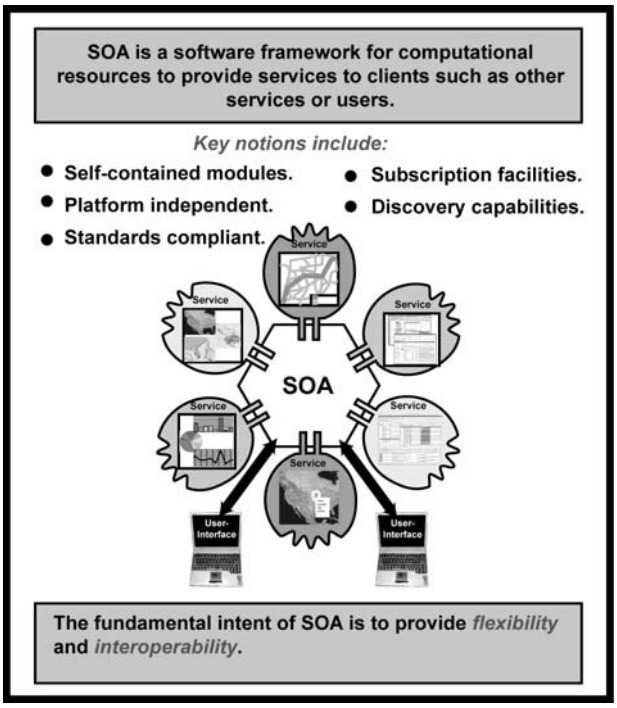

Figure 9: Service-Oriented Architecture (SOA)

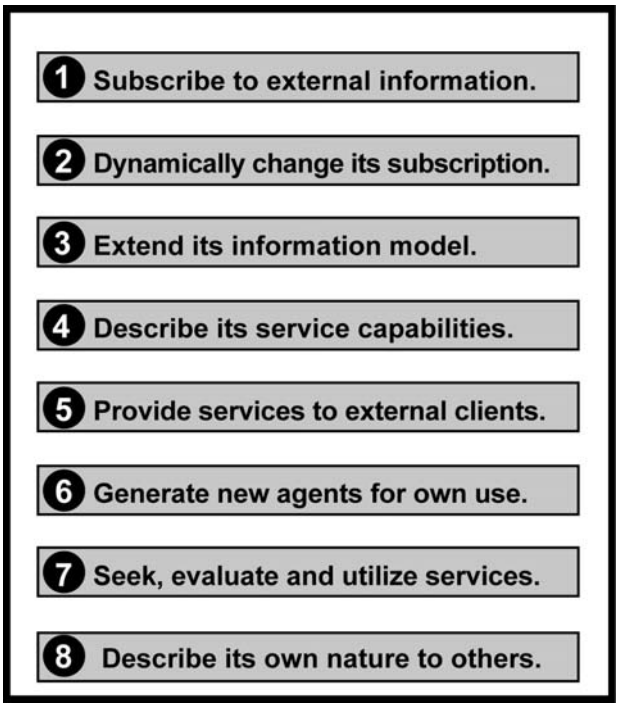

Figure 10: TEGRID capabilities

In the broadest sense SOA is a software framework for computational resources to provide services to customers, such as other services or users. The Organization for the Advancement of Structured Information (OASIS)[1] defines SOA as a *"… paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains"* and *"…provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects with measurable preconditions and expectations"*. This definition underscores the fundamental intent that is embodied in the SOA paradigm, namely *flexibility*. To be as flexible as possible a SOA environment is highly modular, platform independent, compliant with standards, and incorporates mechanisms for identifying, categorizing, provisioning, delivering, and monitoring services (Figure 9).

In such a software environment any individual service can be designed to meet the following technical specifications:

- Self-sufficiency, interoperability, discovery capabilities, and tools with intelligence.

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

- Platform independence with self-installing, self-configuring, and self-scaling capabilities.
- For the more domain-centric services the ability to expose functionality through objectified, domain-centric client interfaces and interact asynchronous with clients.
- Adherence to industry-standard patterns (e.g., JavaBeans, Property Change Management, etc.).
- The ability to operate in terms of application-specific notions and concerns.
- Information-centric representation of *context* to support meaningful human-to-agent and agent-to-agent collaboration.

However, as impressive as these interoperability and functional capabilities may be in comparison with existing legacy systems they represent only the beginning of what is implied by an information-centric system environment. The vision is that of a semantic web 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, demonstrated by the Collaborative Agent Design Research Center (CADRC) during an Office of Naval Research Conference in 2002 (Gollery and Pohl 2002). TEGRID featured several kinds of web service providers, each implementing a set of operations in support of the exchange of the information that was critical to the functioning of the system. These operations included subscription, information transfer, warning and alert generation, discovery, and assignment. Other operations, less critical to the proper functioning of the system, could have been added for real world implementations.

TEGRID utilized a number of standard Internet protocols and elements. These elements were combined into executing software entities 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. Each of these software entities consisted of three principal components: a web server; a semantic web service; and, an information-centric application. The web server utilized standard Hypertext Transfer Protocol (HTTP), serving as the gateway for gaining access to other existing web services[2]. The semantic web service (i.e., a web service with an internal information model) was accessed through the web server utilizing standard protocols (e.g., UDDI, SOAP, WSDL, SML). Its purpose was to provide programmed functionality[3]. The addition of an internal information model in a semantic web service allows the storage of semantic level descriptions

[2] 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.

[3] Clients to a standard web service are usually restricted to those services that implement specific predefined interfaces. However, 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.

(i.e., information) and the performance of limited operations, such as reasoning, on these semantic descriptions. The information-centric applications were 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 and support more sophisticated discovery strategies. In particular, the application component was 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.

Incorporating the three components described above, these TEGRID software entities were 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 (Figure 10). Specifically, they were able to operate as *autonomous* entities and discover the capabilities of other entities. Each entity had a sense of *intent* to accomplish one or more objectives, ranging 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 entities.

Near term and longer term research challenges

While TEGRID did demonstrate the potential feasibility of a fully functional information-centric software environment it also identified capability gaps that call for further research. Attempts to work around these technical shortcomings led to some rather primitive solutions that flawed the overall achievement of the TEGRID demonstration. These included the ability to share portions of the internal knowledge model of a discovered service with the discovering service and the ability of a service to undertake semantic searches.

Extensible Ontologies: Currently the ontologies of information-centric systems are essentially static in nature. In other words, changes and extensions to the information representation structure cannot be implemented dynamically during the execution of an application. Yet, for several reasons it is highly desirable for ontologies to progressively evolve during the operation of information systems. First, this would allow an information system to automatically extend the granularity of a high level core ontology, representing general concepts and notions, into a biased and much more detailed application-specific domain (Figure 11). Second, the ability to dynamically extend an ontology would allow an information system to capture the representation of new objects and relationships and automatically build them into the existing representation structure, thereby dynamically extending the *context* of the decision-making environment within the computer. Third, the dynamic generation of components of an existing information representation structure appears to be a prerequisite for the automatic extraction of information from unstructured data (e.g., free-format text). Fourth, a promising approach for achieving interoperability among multiple applications, at the information level, is based on the concept of a core overarching ontology that is linked to multiple application-specific ontologies, often referred to as facades (Pohl 2001). The latter are viewed as

perspective filters of the core ontology, biased to reflect the native characteristics of a specific application domain. Finally, the ability of a semantic web service to merge part of the ontology of a discovered service with its own internal ontology would be paramount to a low level learning capability (Figure 12).

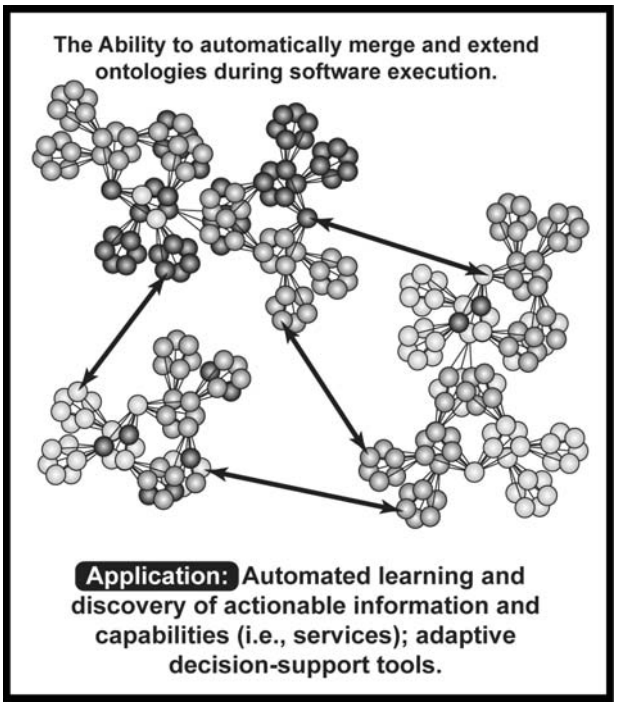

Figure 11: Extensible ontologies

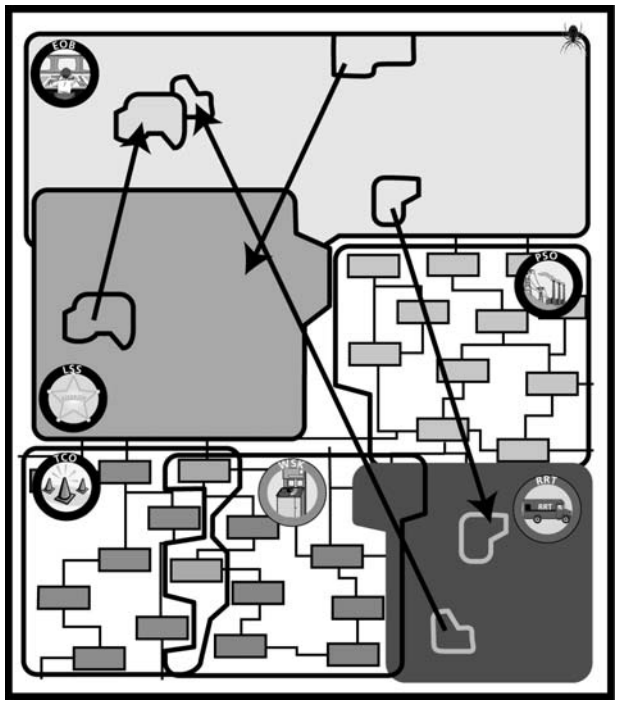

Figure 12: Merging information in TEGRID

Closely associated to the need for dynamically generated ontologies are two related research problems. The first problem deals with the inflexibility of predefined software agents. Typically, the capabilities of software agents are defined at the development stage of an information system. Changes to these capabilities cannot be easily implemented by the user, but normally require the intervention of the software developer. It would be highly desirable for the user or a semantic service to be able to define the capabilities of an agent and have the system automatically create and implement this new agent during normal execution. While some technical capabilities for the dynamic creation of software agents currently exists, these methods are largely limited to predefined functional specifications.

The second problem relates to the capture of information by the system. Ideally, all input should be captured by the system at the point of entry, as information (i.e., within the context of an ontology). In practice, however, much of the input from external sources is in the form of data (e.g., voice recognition, data-centric applications, free text messages, signals, and so on). While several available technologies such case-based classification[4],

[4] Classification techniques inherently concern determining the similarity between objects that share, to varying degrees, a common set of features. Case-based classification works as follows: for a new object or a case to be labeled, a case-based classifier retrieves the most closely matching previously labeled cases from a database of cases, called a *case base*, and assigns the label from the retrieved cases as the label for the new object.

similarity assessment methods[5], and text-based similarity methods have been applied and tested in diverse application domains their combination in a hybrid data interpretation and information fusion system environment requires further research.

Semantic Search Capabilities: The scope of database query facilities desirable for the kind of semantic services envisioned in a TEGRID environment far exceed traditional database management system (DBMS) functions. They presuppose a level of embedded intelligence that has not been available in the past. Some of these desirable features include: conceptual searches instead of factual searches; automatically generated search strategies instead of predetermined search commands; multiple database access instead of single database access; analyzed search results instead of direct (i.e., raw) search results; and, automatic query generation instead of requested searches only (Figure 13).

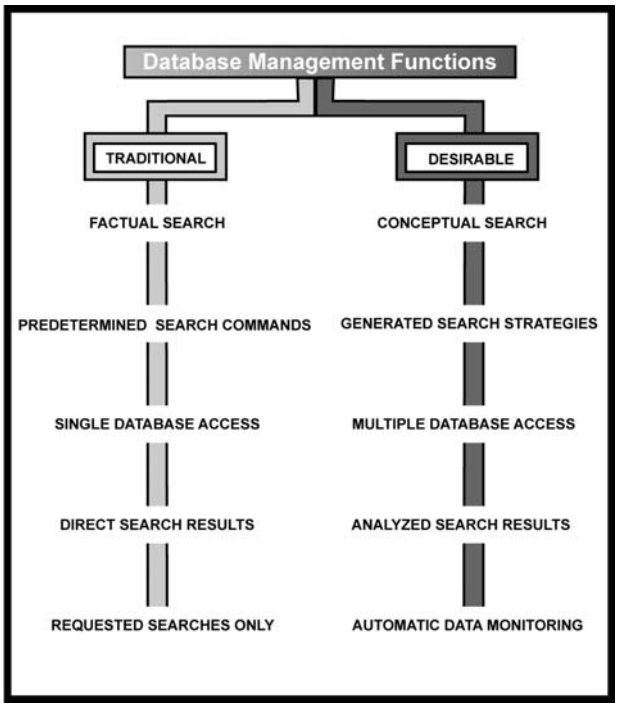

Figure 13: Comparison of directed and semantic search capabilities

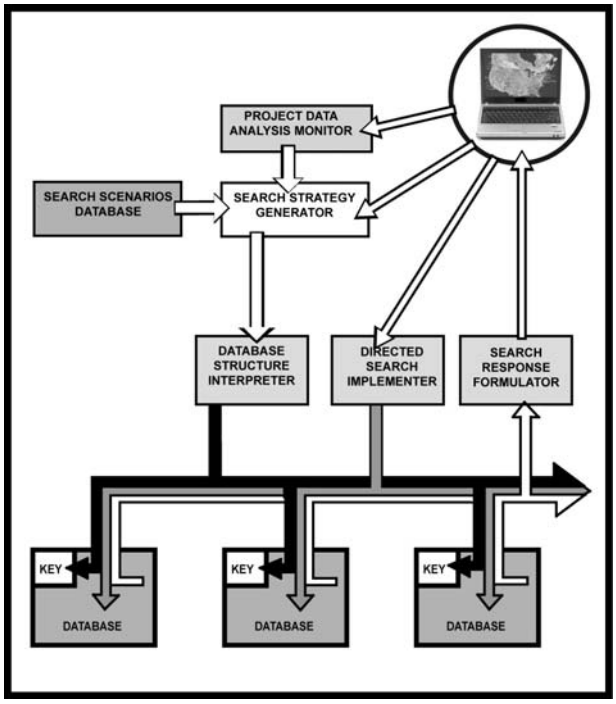

Figure 14: A conceptual semantic search environment

A traditional DBMS typically supports only factual searches. In other words, users and applications must be able to define precisely and without ambiguity what data they require. In complex problem situations users rarely know exactly what information they require. Often they can define in only conceptual terms the kind of information that they are seeking. Also, they would like to be able to rely on the DBMS to automatically broaden the search with a view to *discovering* information.

[5] Classifying elements in a complex and multifaceted domain tends to require the amalgamation of multiple classification methods that each excel in different aspects of similarity assessment. The relative performance of each individual method is domain-specific and often difficult to predict without real-world usage. By wrapping the classification methods as distinct similarity assessment methods, each calculating its own similarity score, domain-specific selection and relative weighting of those methods can be achieved.

This suggests, in the first instance, that an intelligent DBMS should be able to formulate search strategies based on incomplete definitions. It should be able to infer, from rather vague information requests and its own knowledge of the requester and the problem context, a set of executable query procedures. To facilitate this process the DBMS should maintain a history of past information requests, the directed search protocols that it generated in response to these requests, and at least some measure of the relative success of the previous search operation.

A traditional DBMS normally provides access to only a single database. A knowledge-based decision-support environment is likely to involve many information sources, housed in a heterogeneous mixture of distributed databases. Therefore, through the internal-level database representations discussed earlier, the DBMS must be able to access multiple databases. Using the mapping functions that link these internal representations an intelligent DBMS should be capable of formulating the mechanisms required to retrieve the desired data from each source, even though the internal data structures of the sources may differ widely. Particularly when search results are derived from multiple sources and the query requests themselves are vague and conceptual in nature, there is a need for the retrieved information to be reviewed and evaluated before it is presented to the requester. This type of search response formulation facility has not been necessary in a traditional DBMS, where users are required to adhere to predetermined query protocols that are restricted to a single database.

Finally, all of these capabilities (i.e., conceptual searches, dynamic query generation, multiple database access, and search response formulation) must be able to be initiated not only by the user but also by any of the computer-based agents that are currently participating in the decision-making environment. These agents may be involved in any number of tasks that require the import of additional information from external databases into their individual knowledge domains.

A conceptual model of an intelligent DBMS interface with the capabilities described above should be able to support the following typical information search scenario that might occur in an integrated and distributed, collaborative, multi-agent, decision-support environment (Figure 14). Queries that are formulated either by the user or generated automatically by a computer-based agent are channeled to a Search Strategy Generator. The latter will query a Search Scenario Database to determine whether an appropriate search strategy already exists from a previous search. If not, a new search strategy is generated, and also stored in the Search Scenarios Database for future use. The search strategy is sent to the Database Structure Interpreter, which automatically formulates access protocols to all databases that will be involved in the proposed search. The required access and protocol information, together with the search strategy, are sent to the Directed Search Implementer, which conducts the required database searches. The results of the search are sent to a Research Response Formulator, where the raw search results are analyzed, evaluated and combined into an intelligent response to be returned to the originator of the query.

The proposition that the DBMS interface should be able to deal with incomplete search requests warrants further discussion. When searching for information, partial matching is often better than no response. In traditional query systems, a database record either matches a query or it does not. A *flexible* query system, such as the human brain, can

handle inexact queries and provide best guesses and a degree of confidence for how well the available information matches the query (Pohl et al. 1992 and 1994). For example, let us assume that a military commander is searching for a means of trapping a given enemy force in a particular sector of the battlefield and formulates a *something like* a choke point query. In a flexible query system a *something like* operator would provide the opportunity to match in a partial sense, such as: terrain conditions that slow down the movement of troops; unexpected physical obstacles that require the enemy to abruptly change direction; subterfuge that causes enemy confusion; and so on. These conditions can all, to varying extent, represent *something like* a choke point that would be validated by a degree of match qualification.

Flexible query processing systems are fairly common. For example, most automated library systems have some level of subject searching by partial keyword or words allowing users to browse through a variety of related topics. Even word-processing programs include spelling checkers, which by their very nature search for similar or related spellings. However, even a flexible query system cannot automatically form hypotheses, since the system does not know what to ask for.

The ability to search for *something like* is only a starting point. How can the system be prompted to search for vaguely or conceptually related information? For example, how can the system discover the intuitive connection between a physical choke point, such as a narrow cross-corridor in a mountainous battlefield, and a precision fire maneuver aimed at concentrating enemy forces in an exposed area. In other words, how can the system show the commander that the precision fire maneuver option can satisfy the same intent as the cross-corridor option? In addition, the system must not overwhelm the commander with an unmanageable number of such intuitive speculations. To discover knowledge it is necessary to: form a hypothesis; generate some queries; view and analyze the results; perhaps modify the hypothesis and generate new queries; and, repeat this cycle until a pattern emerges. This pattern may then provide insight and advice for intuitive searches. The goal is to automate this process with a *discovery* facility that repeatedly queries the prototype knowledge bases and monitors the reactions and information utilized by the decision-maker, until the required knowledge is discovered.

In addition to these two research challenges that are of immediate near term importance as key enabling capabilities during the current transition to an information-centric software environment, there are several other desirable capabilities that are longer term undertakings because they require major research efforts. These include the ability to extract and store the invariant core component of a solution (e.g., plan, design, strategy) in a way that will allow the complete solution to be automatically regenerated in the future (Hawkins and Blakeslee 2004). Any breakthrough in this area, commonly referred to as *hierarchical temporal memory* is likely to have significant impact on the design and capabilities of future decision-support systems. A second area is the automated interpretation of images. With the increased implementation of surveillance technology (e.g., video cameras) there is an urgent need for software systems that are able to continuously monitor and automatically interpret any significant changes in the images that are being recorded.

Hierarchical Temporal Memory (HTM): There is a tendency for us human beings to succumb to the temptation of believing that the goal we have finally reached is the ultimate solution to the problem that we may have been working on for some time. In

fact, what appear to be solutions to major problems typically turn out to be mere stepping stones in an endless evolutionary sequence of problem solving and increased understanding.

For example, the computer was initially conceived as a high speed numerical calculator. However, this turned out to be really only the beginning of digital computer technology. It was soon realized that the ability to store and process data (i.e., both numeric and textual) is even more important. This led to new hardware and software solutions in the form of greatly increased storage density devices (e.g., disk drives) and formal data management languages (e.g., relational database management systems and the Standard Query Language (SQL)). As the data storage capacities of the new hardware devices have increased from kilobytes to megabytes to gigabytes it has become increasingly clear that we are essentially storing and analyzing data without context. The context is provided by the users who interpret the results of the data analysis within the context of their experience-based knowledge and understandings. As explained at the beginning of this paper, the complete reliance on the human interpretation of the rapidly increasing quantity of data created a bottleneck. To overcome this human bottleneck, methodologies were devised for constructing context models of real world problem situations in software. These context models are in the form of ontologies that provide an information structure that is rich in relationships and allows data to be automatically interpreted within the context provided by the ontology.

Again, ontologies are not an ultimate solution but only a stepping stone in the quest for more intelligent computer software tools and services. It could be suggested that the issue is not only related to the representation of context. Software tools, whether intelligent or not, are largely based on the notion of generating solutions based on the interpretation of data in context. Would it not be more productive to find a way of representing and storing solutions (i.e., designs) that can be rapidly retrieved, instead of computing each design from first principles? Such designs could be operational sequences representing entire solutions or, emulating the functions of the human brain's neocortex, only the essential components that can be later quickly assembled into an entire solution (Hawkins and Blakeslee 2004).

The research challenge is twofold, to find a way of extracting the core components of a design and being able to later automatically reassemble the complete design from the core components. Hawkins (2007) and his colleagues at Numenta[6] have developed the Hierarchical Temporal Memory (HTM) theory and a set of tools to emulate some of what they believe to be the functional capabilities of the neocortex of the human brain. In particular, they see the neocortex to be a hierarchical structure like the roots and trunk of a tree. Sensory stimuli enter at the roots level and are hierarchically assembled into progressively more complex and complete configurations (i.e., patterns or designs) at the trunk level. As shown in Figure 15, Hawkins (2007, 23) explains this concept in terms of the hierarchical assembly of an object (i.e., a dog). At the lowest level the key components are spread among many nodes in a fragmented manner. However, at progressively higher levels these components are assembled into the image of a dog.

[6] Numenta is a California company headquartered in Menlo Park, founded in 2005 by Jeff Hawkins, Donna Dubinsky and Dileep George.

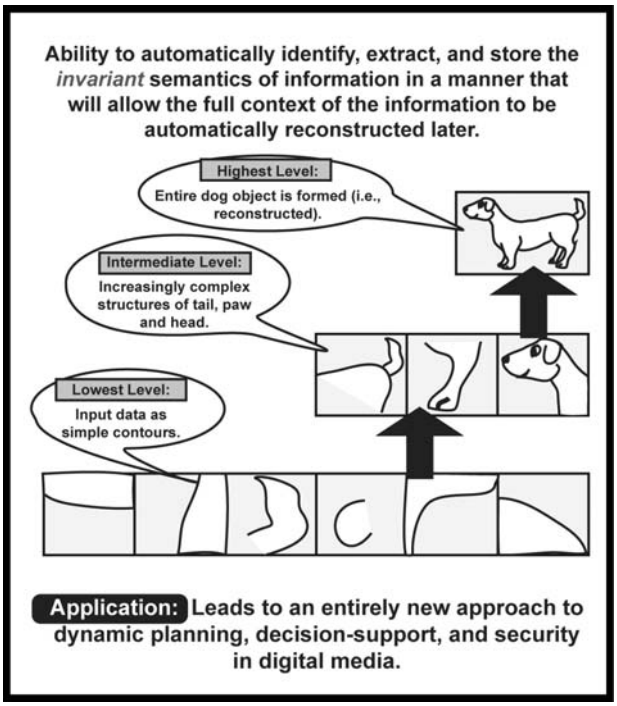

Figure 15: Hierarchical Temporal Memory

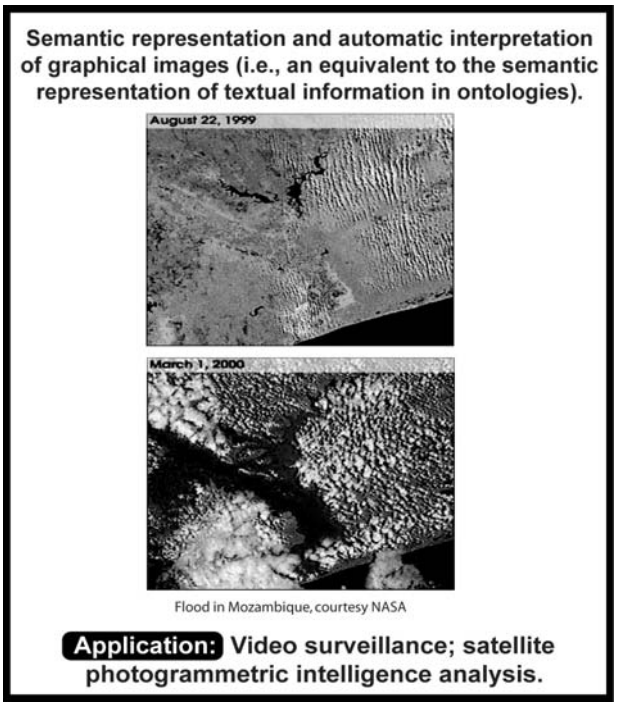

Figure 16: Image interpretation

Specific software research questions that need to be addressed include: What should be the granularity of the partial solution components?; How should the components be assembled?; How can the appropriate components be identified and rapidly retrieved?; How should the solution components be stored?; Will there still be a need for an ontology-like framework to support the rapid identification and retrieval of the components?; and, Should there be a learning component that automatically generates solution components and stores them for future use? A learning capability would certainly be very useful since it would allow the progressive accumulation of a vast knowledge base of partial solution components that can be rapidly adapted and assembled into complete solution.

A reliable HTM capability would have a profound impact on the design of intelligent software tools. Instead of requiring solutions (e.g., a plan) to be developed from the bottom up each time they are required, it would be possible to identify and reassemble an archived past solution. If the solution does not entirely fit the current problem situation it could be modified, much the same way as the human brain modifies prototype solutions and rarely creates a new solution from first principles (Gero et al. 1988, Pohl et al. 1997, 52-55).

Automated Image Interpretation: With the increased emphasis on surveillance and personnel identification there is a need for software tools that are capable of automatically identifying the content of video and graphical images. While much headway has been made in recent years in the development of software that is capable of comparing video clips with archived video images and the application of biometric algorithms for personnel identification, this is not sufficient.

The continuous monitoring of video cameras by human observers is cost prohibitive and

singularly ineffective. Not only do the capabilities of the human cognitive system degrade over time when required to undertake monotonous tasks, but the reliability of human observers under these conditions is questionable. The research challenge is to develop an ontology-like representation that will support the automatic detection and interpretation of changes in video images. The representation should be of sufficient granularity to detect and interpret changes in a scene, beyond the entry or exit of a person or other object.

The capabilities that have been developed to date are largely focused on video recognition technology in which typically an image is converted into a set of attributes, referred to as an image signature. This provides insufficient context for software agents to reason about smaller changes in a scene that could have significant impact on a particular situation such as a hostage or security surveillance setting. It should be possible to reason about image changes at the same level of granularity as is currently possible with textual data in ontology-based software systems.

Conclusions

We are living in one of the most exciting times in human history for very unfortunate reasons. Information technology is advancing at an accelerated rate and has become the enabler of the individual. Global connectivity combined with inexpensive personal computing devices and powerful software tools are allowing a single person to achieve what was a few decades ago the province of an organization comprising many persons. However, the driving forces of these technological advances are of a sinister nature (Pohl 2004). We are facing unpredictable enemies that are forcing governments to impose security measures that are beginning to seriously impact our everyday activities, particularly in the realm of travel.

Apart from these political forces the technical advances themselves are driving the need for further innovation. For example, global connectivity has greatly increased competition in the commercial arena. Today even the most local market place is within easy reach of the most distant potential competitor. Therefore, simply to survive, there is an increasing need for greater efficiency, continuous vigilance, and tools for planning and re-planning in a dynamically changing environment. These tools must be responsive and adaptive. They must be available to the user when needed, be able to exchange data with external sources, and be capable of seamlessly interoperating with other tools and services. Such capabilities require a level of machine intelligence that cannot be achieved with rote data-processing software.

In this paper the author has attempted to define areas in which research challenges exist and the underlying characteristics of human nature that tend to oppose the necessary motivation for pursuing these challenges. While the tensions created in a paradigm shift that is caused by revolutionary changes in technology can be quite severe and slow down the rate of change, history has shown that it will never succeed in preventing the eventual acceptance and exploitation of the new capabilities.

References

Brooks R. (1990); ‘Elephants Don’t Play Chess’; in Maes P. (ed.) Designing Autonomous

Agents, MIT/Elsevier, Cambridge, Massachusetts (pp.3-7).

Cooper S. I. (2002); 'Homeland Security: A View Through the Eyes of Janus'; Office of Naval Research (ONR) Workshop Series on Collaborative Decision-Support Systems'; Proceedings, September 18-19, Quantico, Virginia (pp. 127-138).

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

Gero J., M. Maher and W. Zhang (1988); 'Chunking Structural Design Knowledge as Prototypes'; Working Paper, The Architectural Computing Unit, Department of Architectural and Design Science, University of Sydney, Sydney, Australia.

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

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.

Hawkins J. (2007); 'Why Can't a Computer be More Like a Brain? – Learn Like a Human'; IEEE Spectrum, April (pp. 21-27).

Hawkins J. and D. Blakeslee (2004); 'On Intelligence'; Times Books, Henry Holt and Company, New York, New York.

Holland J. H. (1995); 'Hidden Order: How Adaptation Builds Complexity'; Addison-Wesley, Reading, Massachusetts.

Kauffman S. A. (1992); 'Origins of Order: Self-Organization and Selection in Evolution'; Oxford University Press, Oxford, England.

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

Pohl J. (2004); 'Interoperability and the Need for Intelligent Software'; 6th Office of Naval Research (ONR) Workshop on Collaborative Decision-Support Systems, Quantico, VA, Sep.8-9.

Pohl K. (2001); 'Perspective Filters as a Means for Interoperability Among Information-Centric Decision-Support Systems'; Office of Naval Research (ONR) Workshop hosted by the CAD Research Center in Quantico, VA, June 5-7.

Pohl J. (1999); 'Some Notions of Complex Adaptive Systems and Their Relationship to Our World'; in Pohl J. and T. Fowler (eds.) Advances in Computer-Based and Web-Based Collaborative Systems, focus symposium: International Conference on Systems Research, Informatics and Cybernetics (InterSymp-99), Baden-Baden, Germany, August 2-6, (pp.9-24).

Pohl J., A. Chapman, K. Pohl, J. Primrose and A Wozniak (1997); 'Decision-Support Systems: Notions, Prototypes, and In-Use Applications'; Collaborative Agent Design Research Center, Technical Report CADRU-11-97, Cal Poly, San Luis Obispo, CA 93407 (pp.10-11).

Pohl J., L. Myers and A. Chapman (1994); 'Thoughts on the Evolution of Computer-Assisted Design'; Collaborative Agent Design Research Center, Technical Report CADRU-09-94, Cal Poly, San Luis Obispo, California, September.

Pohl J., J. La Porta, K. Pohl and J. Snyder (1992); 'AEDOT Prototype (1.1): An Implementation of the ICADS Model'; Collaborative Agent Design Research Center, Technical Report CADRU-07-92, Cal Poly, San Luis Obispo, California.

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

Taylor S. (1949); 'Science Past and Present'; Heinemann, London, England (pp.108-129).

Waldrop M. (1992); 'Complexity: The Emerging Science at the Edge of Order and Chaos'; Simon and Schuster, New York.

Wood A. (2001); 'Military Experimentation: Considerations and Applications'; Office of Naval Research Workshop Series on Collaborative Decision-Support Systems'; June 5-7, Quantico, Virginia (pp. 1-8).