CDM Technologies, Inc., 2975 McMillan Avenue, San Luis Obispo, CA 93401 - TEL: 805-541-3750; FAX: 805-541-8296

USTRANSCOM
Corporate Information-Centric Environment (CICE)

Report prepared for the

Architecture & Technical Integration Division

USTRANSCOM TCJ6-A
Scott Air Force Base, Illinois

Report outlines an architecture and proposes an implementation path for extending the existing Corporate Data Environment (CDE) to a Corporate Information-Centric Environment (CICE) that places data in context and provides intelligent software agents in support of an integrated and collaborative knowledge management environment.

October 2003

CDM Technologies, Inc.

2975 McMillan Avenue, Suite 272
San Luis Obispo, CA 93401
(TEL: 805-541-3750; FAX: 805-541-8296; WEB: www.cdmtech.com)

[POC: Jens Pohl (805-756-2841, jpohl@calpoly.edu)]

CDM Technologies, Inc.

2975 McMillan Avenue, Suite 272
San Luis Obispo, CA 93401
(TEL: 805-541-3750; FAX: 805-541-8296; WEB: www.cdmtech.com)

Executive Summary

As a result of foresight and planning that has established a logical data model (i.e., the Master Model), a set of standard reference tables and 17 migration software systems as the foundation of a disciplined Corporate Data Environment (CDE), USTRANSCOM now finds itself in the enviable position of being able to implement DoD's vision of a Global Information Grid (GIG) with speed and confidence. The urgency with which DoD views the need to shift the burden of tedious data filtering and interpretation tasks from human operators to automated machine-based processes, is reflected in the CDE and provides the opportunity for USTRANSCOM to take a leading role in implementing the GIG vision.

USTRANSCOM is inundated with an overwhelming volume of data from a proliferation of heterogeneous internal and external sources. While virtually all of these data are captured and stored electronically, they are largely devoid of context and therefore have to be interpreted and transformed into useful information by human operators, planners and decision makers. The penalties associated with such a human-intensive ***data-centric*** environment include data bottlenecks, aged data, breakdown of data exchange interfaces, and the inability to accurately interpret and analyze data within time-critical constraints. This places USTRANSCOM in a ***reactive*** mode, and forces the expenditure of valuable resources on treating symptoms rather than addressing the core problem.

The report outlines a blueprint for extending USTRANSCOM's existing Corporate Data Environment (CDE) to an ***information-centric*** knowledge management environment that provides a vehicle for making information and knowledge explicit and accessible throughout the organization. The ***Corporate Information-Centric Environment (CICE)*** architecture is proposed as a core solution to support: (1) the automatic filtering of data by placing data into information context; (2) the automated reasoning of software agents as they monitor events and assist human users in planning, replanning and decision-making tasks; and, (3) autonomic computing capabilities that allow systems to monitor, diagnose and troubleshoot themselves. The principal objective of CICE is to shift the burden of lower level data interpretation, filtering, and context building, from human operators to automated software-based processes.

The proposed CICE architecture consists of four layers, namely: a ***Data Source Layer*** of external and internal applications and databases; a ***Data Integration Layer*** that cleanses, integrates, and stores the data in an Operational Data Store and a Data Warehouse with several focused Data Marts; a ***Mediation Layer*** that provides metadata registries, reference data tables and a logical data model (i.e., the Master Model) to facilitate the discovery of data within the appropriate context; and, an ***Information Management Layer*** that includes ontology-based applications with intelligent, collaborative tools (i.e., software agents). These are the essential layers of a fully capable knowledge management environment that shifts the burden of data filtration and interpretation from valuable and scarce human resources to the machine.

The CICE implementation path builds on key components of the existing CDE, which are relocated to the appropriate layer within the proposed four-layer architecture. These components include: the Operational Data Store and Data Warehouse with data Marts, which are moved into the Data Integration Layer; the Master Model, metadata registries, CRIS database, and standard reference tables, which are moved into the Mediation Layer; and, the migration systems, which are moved into the Data Source Layer. The existence of these components together with a

disciplined data management process (i.e., standardized nomenclature, logical data model, principles, and compliance evaluation procedures) have favorably positioned USTRANSCOM to proceed expeditiously with the implementation of the CICE architecture.

The report outlines a set of good practices to guide the implementation of CICE and the design of its principal components. These guiding principles are intended to provide a starting point for the disciplined development of ontologies and ontology-based applications, as well as to ensure that the implemented form of the CICE architecture is scalable, flexible, and amenable to future technological advances. In particular, attention is drawn to the importance of involving the users of the new intelligent tools that will be provided by CICE, in the design and evaluation of these tools. It is suggested that a use-case centered approach be adopted in support of an iterative software development process, which delivers functional software at short intervals to end-users who then provide feedback and guidance on future requirements. Experience has shown that such an approach facilitates the precepts and objectives of the ***spiral software development*** model.

The report concludes with a set of recommendations that are intended to facilitate the progressive implementation of CICE and provide a sound foundation for the expected growth of such an enterprise-wide knowledge management environment over the next decade. As a first step it is recommended that the CICE concepts, principles and architecture be incorporated in the Defense Transportation System Enterprise Architecture. Concurrently with this largely documentation effort, it is further recommended that CICE implementation proceed along three parallel paths. The first path is technical in nature and focuses on the evaluation of the existing metadata components of the CDE in respect to their ability to support the requirements of CICE. These recommendations define the need for technical investigations and propose some exploratory development efforts to provide certain basic tools that are known to be needed in the Mediation Layer. The second path recommends the formation of a coordination and advisory committee structure with both technical and operational (i.e., functional) representation to provide oversight and ensure organizational acceptance and participation. The third path suggests early planning and consideration of the impact of CICE on current software development projects and contracts, with a view to developing a strategy for aligning these existing and planned efforts with the CICE initiative.

CDM Technologies, Inc., 2975 McMillan Avenue, San Luis Obispo, CA 93401 - TEL: 805-541-3750; FAX: 805-541-8296

Table of Contents

CDM Technologies, Inc., 2975 McMillan Avenue, San Luis Obispo, CA 93401 - TEL: 805-541-3750; FAX: 805-541-8296

CDM Technologies, Inc., 2975 McMillan Avenue, San Luis Obispo, CA 93401 - TEL: 805-541-3750; FAX: 805-541-8296

1. Purpose of Report and Introduction

The purpose of this report is threefold. First, to define an architecture that extends the existing USTRANSCOM Corporate Data Environment (CDE) to an information management facility that is capable of providing the right information to the right people at the right time, within an intelligent collaborative decision-support environment. Second, to suggest steps that could be taken immediately to improve, consolidate and extend the CDE. And third, to explain the technical concepts and notions of the appropriate information technology in largely non-technical terms. These explanations are mostly confined to Section 2 and are intended for readers who do not have a deeper technical knowledge of information technology.

Referred to in this report as the Corporate Information-Centric Environment (CICE), it is the goal of CICE to greatly increase the ability of USTRANSCOM commanders and staff to expeditiously and efficiently plan, execute, monitor, and replan, the movement and distribution of military goods. Consonant to the principles defined by the Department of Defense (DoD) Net-Centric Data Strategy (Stenbit 2003), CICE will allow USTRANSCOM planners, decision-makers, and operators to progressively leverage and exploit the advances in information technology that are becoming available at an increasing rate. Specifically, the CICE architecture and its guiding principles, are designed to support the incremental implementation of progressively more intelligent and powerful tools operating within a collaborative decision-support environment that alerts its users to unforeseen events, assesses risks, identifies opportunities, generates alternatives, maintains in-transit visibility, facilitates time-critical replanning, and collects lessons learned.

1.1 Technical Underpinnings

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 that 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.

CDM Technologies, Inc., 2975 McMillan Avenue, San Luis Obispo, CA 93401 - TEL: 805-541-3750; FAX: 805-541-8296

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

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

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

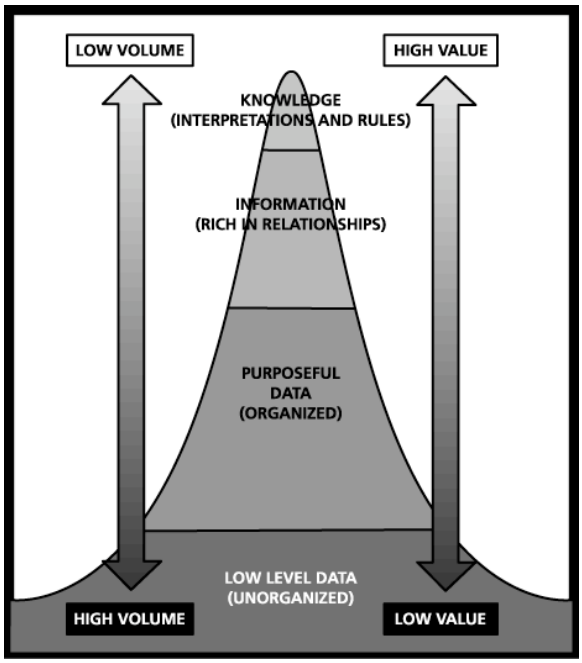


Figure 1: Transition from data to knowledge

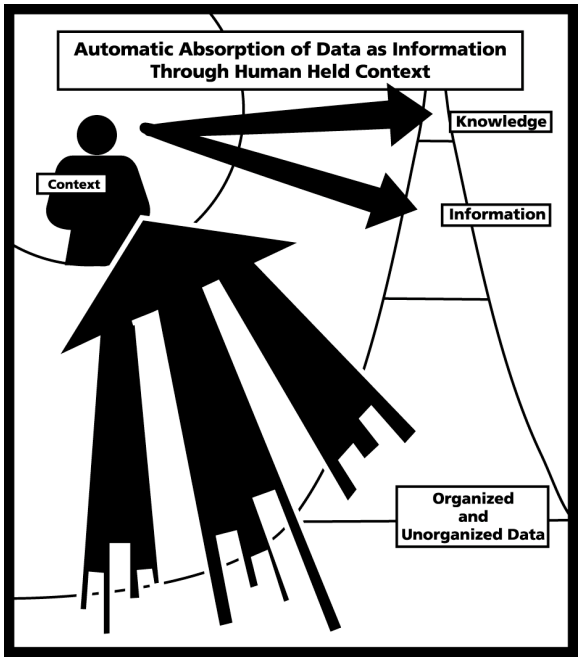


Figure 2: Human interpretation of data

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.

1.2 Transformational Forces

The transformational forces that are driving the need for more intelligent and useful information management environments are related to: (1) progressively scarcer human resources; (2) economic pressures that necessitate more integrated and tighter fiscal control; (3) increased tempo of military missions; (4) greater emphasis on flexibility and adaptability; (5) increased expectations of accuracy; and (6) the overriding necessity for a high degree of interoperability to ensure the consideration of all relevant aspects of a particular planning or problem solving situation.

Increasingly in recent years the ability of the Defense Transportation System (DTS) to support the full range of USTRANSCOM responsibilities and functions with the required speed, quality, and convenience, has been severely tested. The ‘Transformation Study Report: Transforming

Military Operational Cap Capabilities' (Office of the Secretary of Defense, April 2001), Joint Vision 2020 (CJCS 2000), the 'Agile Transportation for the 21st Century (AT-21)' program, and the 'Defense Transportation System – Enterprise Architecture' initiative, all address the urgent need for a global transportation system facility that ensures: ***information accessibility***; ***rapid decision capability***; ***reliability***, ***security*** and ***force protection***; ***in-transit visibility*** and ***adaptability***; ***system interoperability***; and, ***graceful degradation***.

These performance objectives are applicable within each of USTRANSCOM's three key areas: ***financial controls*** (maintaining real-time visibility of financial status and the ability to maximize returns to stakeholders); ***organizational controls*** (shaping the workforce and structure in response to customer needs and market conditions); and, ***process controls*** (applying information processes and business rules to achieve optimum effectiveness and efficiency). Since information technology (IT) is the core capability and principal enabler that supports virtually all functional domains within these key areas, it should be recognized that transformation activities undertaken by USTRANSCOM to improve the performance of DTS must focus foremost on the adoption and implementation of the appropriate information technology approach.

Concurrently, for the past several years the military services have been confronted with societal, political and technological changes that have and will continue to have a profound influence on the manner in which they must plan and execute their missions, and on the ability of USTRANSCOM and its component commands to effectively support the attendant transportation requirements. Prominent among these changes are the following:

> ***Expeditionary Warfare:*** Increasingly, the US military forces have been based in the Continental United States (CONUS), although most of their missions are taking place outside CONUS. This places a great deal of emphasis on rapid deployment capabilities, as well as integrated, reliable communication facilities and seamless coordination capabilities.
>
> ***Joint Task Forces:*** Increasingly, the US military forces are involved in joint operations, often involving foreign countries as allies. This requires an unprecedented level of coordination, flexibility, interconnectivity, global sensitivity and knowledge (e.g., foreign languages), training, and standardization.
>
> ***Force Protection:*** Increasingly, the value of human life is becoming a major factor in US military operations that are under the scrutiny of continuous media coverage. Casualties are a liability with severe political repercussions. Effective protection of the warfighter has become a difficult and expensive proposition.
>
> ***Smaller Unit Operations:*** By far the majority of missions, whether humanitarian or military, are involving smaller forces that may be widely dispersed. Effective, fail-proof communication, real-time coordination, and computer-based, intelligent decision-support are essential prerequisites for such missions.
>
> ***Functional Integration:*** The increased tempo of military operations is requiring planning, rehearsal, execution, re-planning, and training to be viewed as integral functions that must be supported in one integrated communication and coordination system. The traditional separation of these functions into distinct systems is expensive, fragmented, time-consuming, and inefficient.

CDM Technologies, Inc., 2975 McMillan Avenue, San Luis Obispo, CA 93401 - TEL: 805-541-3750; FAX: 805-541-8296

In combination, these changes have added a degree of time criticality in responding to asymmetric threats and a level of decision-making complexity in a largely distributed command and control environment that mandates the availability of computer-based decision-support systems. However, for such decision-support systems to be effective they must have intelligent, collaborative, adaptive, and seamless interoperability capabilities.

CDM Technologies, Inc., 2975 McMillan Avenue, San Luis Obispo, CA 93401 - TEL: 805-541-3750; FAX: 805-541-8296

(This page is intentionally left blank.)

2. Knowledge Management Concepts

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 reasons for this concern are due to two interrelated factors, namely advances in information technology and increasing performance expectations. Technical advances in the processing and storage capacity of digital computers, together with the linkage of these computers into networks of distributed nodes, have greatly increased the capability of organizations to deliver goods and services. With these increased capabilities have come heightened expectations for quality, accuracy, responsiveness, and capacity.

To meet increasingly more exacting performance requirements, organizations have been forced to frequently reexamine their formal structure, processes, and the ability of its members to collaboratively and expeditiously perform their tasks. Since these tasks depend largely on the knowledge held by individuals, the need to share this implicit knowledge explicitly among groups has become more and more important. Under these circumstances it is natural for organizations to take steps to protect their knowledge assets, by collecting and storing the knowledge of individuals in an explicit form that will make it readily accessible to others. This requirement cannot be satisfied by the storage of data only. In addition to data, the explicit representation of knowledge requires the storage of the mappings that convert data into information to place data into context, and the rules (i.e., business rules) that allow information to be effectively utilized in planning, problem solving, and decision-making processes.

A key characteristic of knowledge is the ability to abstract information and relationships so that they can be applied to problems that are outside of the realm of a person's existing experience. In this respect ***abstraction*** is a very powerful representational mechanism that allows for the generalization (not to be confused with vagueness as these generalizations are often quite concrete) of basic and sometimes common characteristics.

2.1 Knowledge as a Commodity

As information technology permeates all aspects of life and the economy turns decidedly information-centric, wealth is increasingly defined in terms of information-related services and the availability of knowledge. In other words, knowledge has become a commodity that has value far in excess of the manufactured products that represented the yardstick of wealth during the industrial age.

How this new form of human wealth should be effectively utilized and nurtured in commercial and government organizations has in recent years become a major preoccupation of management. The question being asked is: How can we capture and utilize the potentially

available knowledge for the benefit of the organization? The phrase "…potentially available" is appropriate, because much of the knowledge is hidden in an overwhelming volume of computer-based data. What is not commonly understood is that the overwhelming nature of the stored data is due to current processing methods rather than necessity. These processing methods have to rely largely on manual tasks because only the human user can provide the necessary context for interpreting the computer-stored data into information and knowledge. If it were possible to capture information (i.e., data with relationships), rather than data, at the point of entry into the computer then there would be sufficient context for computer software to process the information automatically into knowledge. ***This is not just a desirable capability, but an absolute requirement for the capture and effective utilization of knowledge within an organization such as USTRANSCOM.*** Corporate knowledge also acts as an extension to individual knowledge. The representation and storage of 'best practices', combined with proper navigational tools, contributes to the efficiency of individuals performing tasks for which they may not have sufficient experience.

2.2 Information-Centric Computer Software

Apart from an organizational structure that encourages initiative and self-determination, and leadership that provides vision and guidance, there is a third prerequisite for a successful knowledge management environment. This prerequisite is related to the capture and exploitation of the information and knowledge that is generated within an organization. What is the nature and form of this information? It includes not only the continuous information streams with external parties such as e-mail messages, telephone calls, minutes of business meetings, and other documents, but also the information and knowledge that is generated within the organization. The latter is typically fragmented throughout the organization and much of it is potentially lost soon after it has been created and used for a particular purpose. It ranges from the minutes of internal meetings, proposals, reports, white papers, technical references, to the cumulative experience and knowledge that resides in the memory of the members of the organization. In most existing organizations attempts to capture this information vary from formal systematic efforts such as maintaining an on-line database of customer service calls and response actions, to some nebulous knowledge of who worked on a particular project and might therefore be able to contribute some key information to the current problem.

With the increasing realization that the information and knowledge generated through the internal and external activities of an organization constitutes a major asset and must therefore be a key component of any knowledge management plan, many organizations are asking themselves the following questions: What are the fundamental elements of this resource? How can this resource be efficiently captured at the source and stored electronically? Does this resource have to be processed (e.g., validated, analyzed, and evaluated) in some way to make it useful? and, How can we provide convenient access and yet keep this valuable resource secure? These questions form the focus of the following sections.

The fundamental elements: The principal elements or building blocks of a knowledge management system are data, information, knowledge, and wisdom (Figures 3 and 4). Data essentially are numbers and words without relationships. We human beings are able to interpret data into information by utilizing the context that we have accumulated in our cognitive system over time (i.e., our experience). Computers do not have a human-like cognitive system and

therefore any data stored in a computer will need to be interpreted by the human user. While the computer is able to order, recast, categorize, catalog, and process the data in many different ways, it cannot use them as the basis of any reasoning sequence. However, if we store not only the data but also a rich set of relationships that place the data into context then it is not difficult to develop software modules (i.e., agents) with reasoning capabilities. In this way it is possible to develop software with some internal understanding of what it is processing (Pohl 2001).

The ability to represent information in computer software has been available for at least the past 30 years (Winston 1970, Biermann and Feldman 1972, Cohen and Sammut 1978). Hampered initially by a lack of hardware power and later by the absence of any compelling need to involve the computer in the direct interpretation of data, these information modeling techniques were not applied in the mainstream of computer software development until fairly recently. The compelling reasons that have suddenly brought them to the foreground are the increasing volume of computer-based data that is beginning to overwhelm human users, and the homeland security concerns that emerged after the tragic September 11, 2001 terrorist incidents in the United States.

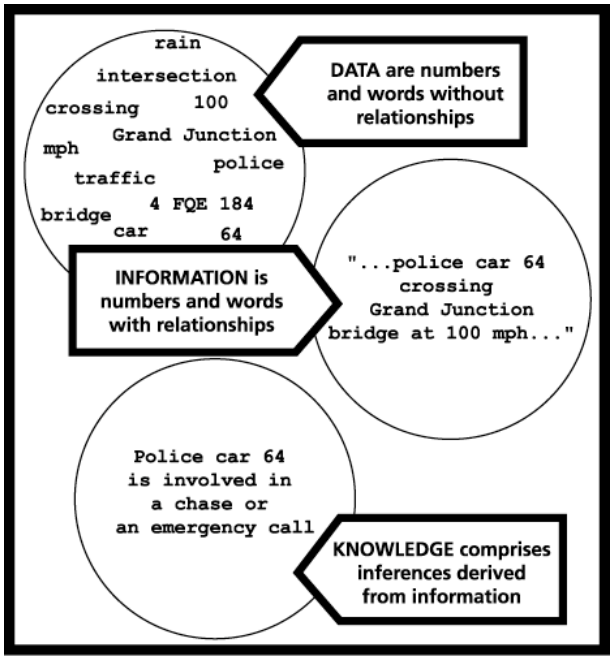


Figure 3: Data, information and knowledge

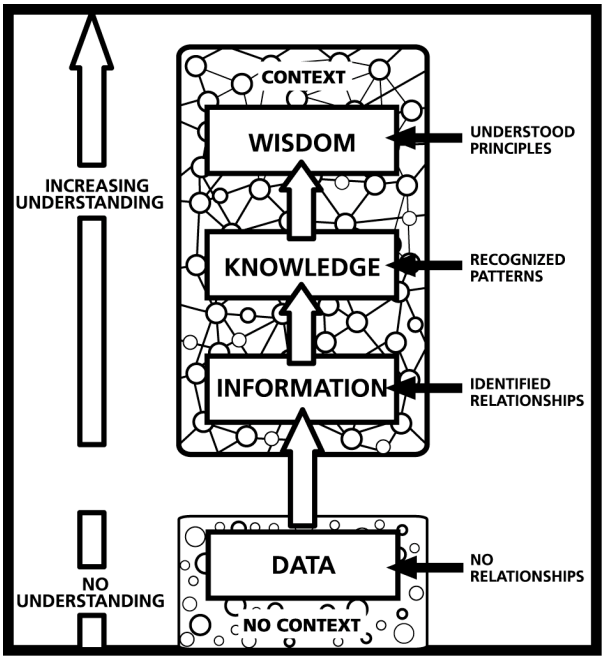


Figure 4: Importance of context

The physical gap that is shown schematically between the realms of the data environment without context and no understanding, and the information environment with context and ascending levels of greater understanding in Figure 4, is intended to underscore the fundamental difference between the two realms. The transition from data-processing software to information-centric software requires a paradigm shift in the human perception of the role of computers. By incorporating an internal information model (i.e., ontology) that represents portions of real world context as a virtual environment of objects their characteristics and the associations that relate these objects, information-centric software is capable of performing a useful level of automatic reasoning. A number of software agents with relatively simple reasoning capabilities are able to collaborate and through their collective efforts come to more sophisticated conclusions.

CDM Technologies, Inc., 2975 McMillan Avenue, San Luis Obispo, CA 93401 - TEL: 805-541-3750; FAX: 805-541-8296

The architecture of a knowledge management system: 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 but not always, with summarized data collected from operational data sources. In some cases, for example when the operational data sources are particularly fragmented and unreliable, a Data Warehouse may serve as an operational data store that cleans and integrates data at the same granularity as they exist in the original 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. 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.

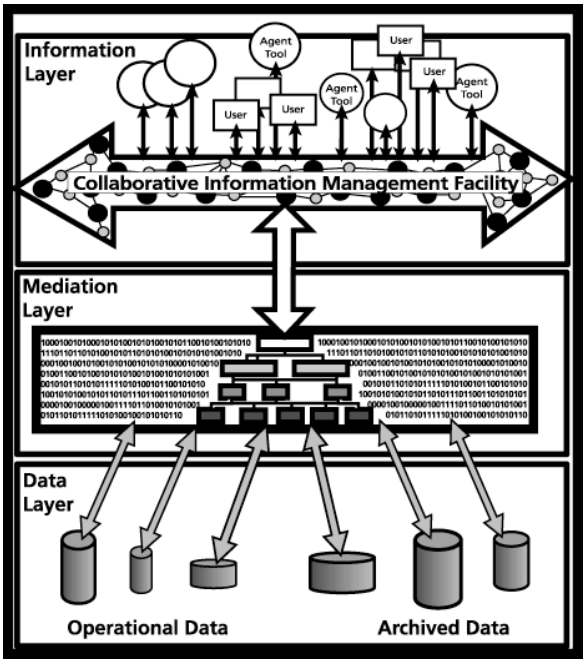


Figure 5: Schematic architecture of a knowledge management system

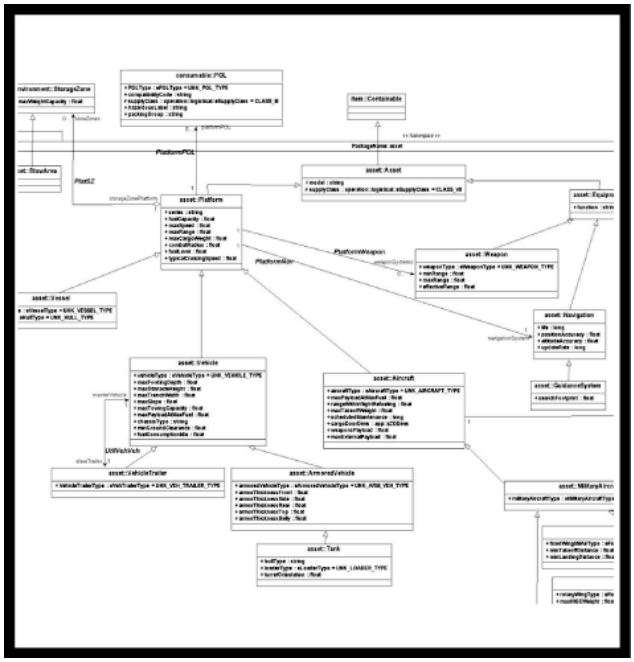


Figure 6: Portion of a typical information model (ontology) in the logistic domain

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 5). The upper layer utilizes an internal information model (i.e., ontology, Figure 6) to provide context for the automatic reasoning capabilities of software agents. Essentially, these agents enabled by their reasoning capabilities constitute a set of intelligent tools that continuously monitor the events (i.e., changes) occurring in the operational environment.

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 (Figure 5). In this manner, the ontology of the information management 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.

Intelligent information management tools: There are many types of software agents, ranging from those that emulate symbolic reasoning by processing rules, to highly mathematical pattern matching neural networks (McClelland and Rumelhart 1988), genetic algorithms (Koza 1992), and particle swarm optimization techniques (Kennedy and Eberhart 2001). In general terms software agents are defined by Wooldridge and Jennings (1995) as "… *computer systems, situated in some environment, that are capable of flexible autonomous actions* …". The three critical words in this definition are situated, flexible, and autonomous. Situated means that the agent receives information from its environment and is capable of performing acts that change this environment. Autonomous refers to the agent's ability to act without the direct intervention of human users. In other words that the agent has some degree of control over its own actions and internal state. And, flexible means that the system is: responsive - by perceiving its environment and being able to respond in a timely fashion to changes that occur in it; proactive - by exhibiting opportunistic, goal-directed behavior and exercising initiative where appropriate; and, social - by interacting, when appropriate, with other agents and human users in order to complete its own problem solving tasks and help others with their activities.

How do these characteristics of software agents translate to the kind of knowledge management system described above (Figure 5)? The agent tools are situated since they receive a continuous flow of operational information generated by the activities of the organization, and perform acts that may change that environment (e.g., creating alerts, making suggestions, and formulating recommendations). The agent tools are autonomous because they act without the direct intervention of human users, even though they allow the latter to interact with them at any time. In respect to flexibility, the agent tools possess the three qualities that define flexibility within the context of the above definition. They are responsive, since they perceive their environment through an internal information model (i.e., ontology) that describes many of the relationships and associations that exist in the real world environment. They are proactive because they can take the initiative in making suggestions or recommendations (e.g., transportation mode selection

for a particular shipment, emergency team configurations in crisis management situations, or route selection for moving troops or equipment) and they do that in an opportunistic fashion. For example, when an emergency call is initiated, a Route agent may immediately and without any explicit request from the user, determine the optimum route under current traffic conditions that should be used by the ambulance to reach the injured person.

The ability of software agents to communicate (i.e., socialize) with each other and with human users to work on their own problems or assist others with their problems, is a powerful capability of the information layer in a knowledge management system. It allows several agents to collaborate and concurrently explore different aspects of a problem from multiple points of view, or develop alternative solutions for future negotiation.

Symbolic reasoning agents that are quite common in knowledge management systems incorporate collections of rules that monitor specific conditions and generate alerts when these conditions are satisfied. The general design of such an agent consists of three components: the conditions that trigger the agent (i.e., the functional specification of the agent); the objects and their attributes that are involved in these conditions (i.e., the part of the internal information model (i.e., ontology) that is used by the agent); and, the logic that defines the relationships among these objects and attributes.

One important aspect of autonomy in agent applications is the ability of agents to perform tasks whenever these may be appropriate. This requires agents to be continuously looking for an opportunity to execute. In this context opportunity is typically defined by the existence of sufficient information. For example, to identify a shortage of inventory either some agent has to monitor the consumption of the particular inventory item until there is a shortage and then issue a warning, or one or more agents collaboratively project that based on developing conditions there is likely to be a shortage of the given item at some specific time in the future.

The requirements for rule-based agents are defined in terms of two elements: conditions; and, actions. The conditions are the specifications of the situation that the agent monitors, while the actions are the alerts that should be generated when these conditions are true. Typically, conditions are specified in terms of objects, attributes and the relationships among them. Each condition is formed by a pattern of object, attributes, values, and Boolean tests. Patterns are grouped by logical connectors, such as AND, OR, and NOT. The more patterns and relationships that are specified, the more specific these conditions become. The right hand side of a rule represents the actions to be taken when the conditions are satisfied. The most general type of action is to generate an alert. However, there are many other kinds of actions that rule-based agents can perform (e.g., look for additional information, modify an existing schedule or generate a new schedule, develop a particular solution approach, simulate the likely outcome of a course of action, and so on).

Software agents as intelligent decision-assistance tools: There are high expectations that intelligent software agents will solve many of our current information system woes, such as lack of interoperability, multiple failure points, vulnerability to intrusion, making the right information available to the right person at the right time, and proposing solutions under time-critical conditions. Software agents do not have magical human-like capabilities. It is not possible to simply develop a piece of software code that is capable of reasoning about conditions and circumstances like we human beings appear to be able to do. Computers are not human

beings and definitely do not have human capabilities. Yet, it is indeed possible to develop software agents that are capable of accomplishing human-like tasks such as recognizing certain conditions, reasoning about these conditions, forming conclusions, and taking actions on the basis of those conclusions.

At first sight the above statements may appear to be contradictory. Software agents do not have human-like capabilities and yet, they are able to accomplish human-like tasks. There is obviously a missing link, a particular ingredient in a software environment that makes it possible for software agents to perform tasks that would normally require human intelligence. Although there is an increasing acceptance of the notion of intelligent computer-based agents, what is not generally understood are the kinds of fundamental capabilities that allow such agents to perform intelligent tasks and the nature of the software environment that is required to support these capabilities?

First we should ask ourselves: What precisely are the capabilities that a software agent needs to have to be able, for example, to determine the information required by a given computer user at any point in time and to prepare alternative solutions for a particular problem situation? Clearly, such tasks require ***reasoning*** capabilities. The obvious next question then becomes: How can we make it possible for a piece of software code to reason about anything?

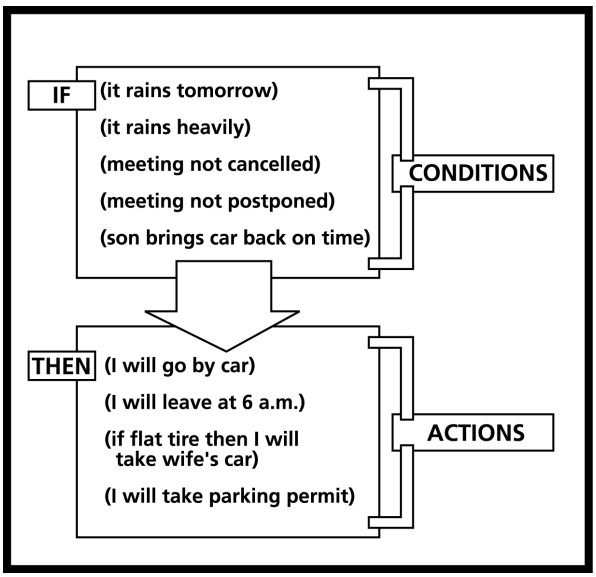


Figure 7: Typical rule (or production)

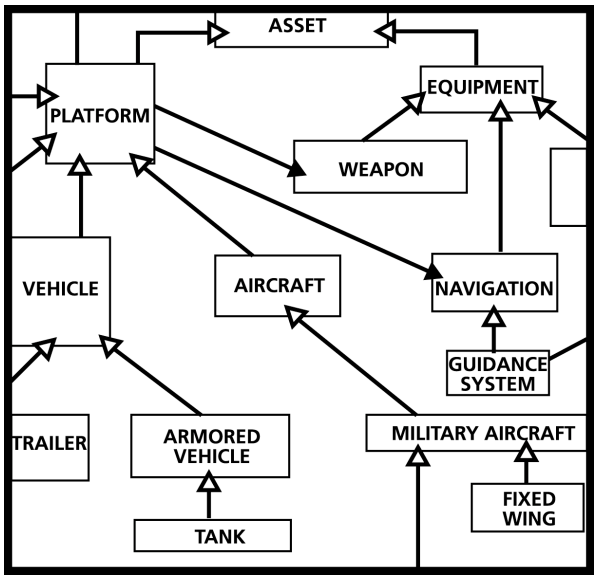


Figure 8: Small part of an ontology

To answer this second question we need to examine in some detail what is meant by reasoning. In general terms, reasoning is a logical process involving the systematic interpretation of information. From our earliest school years we learn to assemble information into a form that facilitates various problem-solving activities. For example, we learn how to extract the few contextually important pieces of information from a passage of text, or how to rearrange a set of conditions to establish a suitable framework for drawing conclusions. In simplest terms this logical process can often be reduced to a set of conditions, the application of certain tests to these conditions, and the drawing of conclusions based on the outcome of the tests. For example (Figure 7): I usually commute to work by bicycle. Tomorrow morning I have to be in the office

very early for a meeting at 7 am. IF it rains tomorrow morning THEN I will not commute by bicycle, but use my car instead. IF I have to use my car THEN I will need to leave 20 minutes earlier than normal to be able to find a parking space, and so on.

Many years before computers became available this process of deductive reasoning was already well understood. Emil Post (1943) coined the term ***productions*** to describe sequences of IF…THEN conditions and conclusions. More familiar to the layperson is the term ***rules***. The IF-part (or predicate) of a rule establishes the conditions that must be satisfied before the THEN-part (or consequent) can be assumed to logically follow. As shown in Figure 7, a single rule may contain multiple conditions and/or actions. In addition, secondary conditions can be embedded in both the IF-part and the THEN-part of a rule.

Rules are of course not the only way in which we can structure problem conditions and a solution sequence. For example, a neural network consisting of interconnected layers of input, intermediate and output nodes utilizes an entirely different approach for detecting a pattern of conditions. It essentially implements a sophisticated mathematical function to generate a very primitive numerical output (i.e., a set of decimal values between 0 and 1) to indicate that a particular input (represented in the same numerical manner) is similar to an input pattern that it has been mathematically trained to recognize. This is quite different from the approach that the rule shown in Figure 7 follows to logically define the conditions that must be met before any of the actions can take place. Instead, the neural network relies on a pattern matching approach that does not require an understanding of the meaning of the recognized patterns but simply the ability to recognize it. Furthermore, additional interpretation has to be provided by other means to convert the real world pattern into an abstract set of numerical values that are fed into the input nodes and convert the numerical output code generated by the neural network into a meaningful (real world) result. Neural networks are powerful tools, even though they do not rely on symbolic reasoning capabilities.

Software agents that are able to analyze problem situations, dynamically changing conditions, and events, must have some understanding of the meaning of the information that they are reasoning about. The question then becomes: How can we create a computer-based environment that conveys to a software agent sufficient meaning for that agent to undertake reasoning tasks of the kind exemplified by the rule shown in Figure 7?

To answer this question it is necessary to reiterate the previous brief reference to the distinction between data and information (see Section 1.1). Data are simply numbers and words, while information adds to data additional very important concepts, abstractions and contextual ***relationships***. The addition of concepts, abstractions and contextual relationships is critical to any reasoning process because they provide ***context***. Without this context even a human being would have great difficulty making sense out of a bunch of data. What makes it so easy for us human beings to reason about a wide range of data is the context that we have accumulated in our cognitive system over time through an experience-based learning process. We automatically convert data to information as long as we can find in our memory the context within which the words and numbers (i.e., data) that our eyes see, convey meaning. In other words, subject to the existence of relevant experience our cognitive system automatically adds the relationships (i.e., context) that are necessary for us to reason about the data.

Furthermore, it is through abstraction that we are able to apply our experience to a wide range of problems, even those that fall outside our specific experience base. This is partially due to the

ability for these abstractions to essentially apply to concepts or entities yet to be incorporated. For example, abstracting the notion of 'trackability' (and all the semantics that accompany such a concept) not only supports currently utilized entities (described as specializations of things that are 'trackable') but at the same time lays a solid foundation for incorporating additional entities that may also embody the property of being 'trackable'.

Since these human cognitive processes are automatic, it is perhaps not unreasonable for us to forget that computers do not have this capability because they do not have an equivalent cognitive system. The same would apply if we were to ask a literate six-year old child to interpret the meaning of a typical printed, single-page agenda of a business meeting. Although the child may be able to readily read the agenda she is unable to make much sense of its contents because she has no prior experience of such meetings. In other words, the child lacks the context that is necessary for reasoning about the agenda.

For the computer to be able to support automatic reasoning capabilities we have to create a software environment that incorporates context. This can be achieved fairly easily by constructing an information model as a virtual representation of the real world context within which software agents are expected to apply their reasoning capabilities. Such an internal information model is referred to as an ontology. A small part of a typical example of such an ontology is shown in Figure 8. It describes the real world context in terms of objects with characteristics and relationships. For example, in a military command and control context such objects would include different kinds of weapons, a wide range of infrastructure objects, weather forecasts, friendly and enemy units, and even conceptual objects such as the notions of threat, planning, mobility, and readiness. Generally speaking, the more relationships among objects that are included in the ontology the more context is provided by the ontology, and the more powerful (i.e., intelligent) the reasoning capabilities of the software agents are likely to be.

Without the context provided by an internal information model (i.e., ontology) there can be no meaningful, automatic reasoning by software agents. Of course there could still be neural network agents and software modules that simply manipulate data based on some predefined data-processing scheme, but neither of these are capable of the kind of symbolic reasoning that is now being referred to under the title of ***intelligent agents***. Therefore, the 'missing link' or essential prerequisite for intelligent agents is the existence of an internal information model that provides the necessary context for the symbolic reasoning activities of the agents. We human beings do not have to consciously invoke any action to relate what we see, hear and feel to the context held in our brain. The need for this context to be created in the computer is therefore not intuitively obvious to us. This is no doubt the principal reason why such a fundamental aspect of intelligent computer-based agents is still largely overlooked.

The Web-Services environment: A knowledge management system may be implemented as a set of Web-Services on the Internet or in any intranet environment (Figure 9). Existing Web-Services environments 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 (Figure 10). 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 would be necessary when passing data between two data-centric applications (Figure 11). In the case of an information-centric Web-Service a data-to-information translation is performed when the Web-Service receives data from an external source and an information-to-data translation is performed whenever the Web-Service sends information through the Web Server (Figure 12).

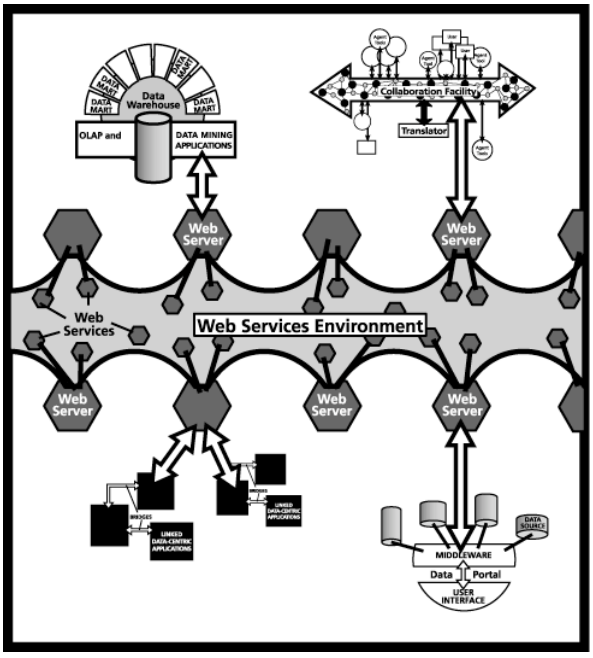


Figure 9: Integration of heterogeneous systems in a Web-Services environment

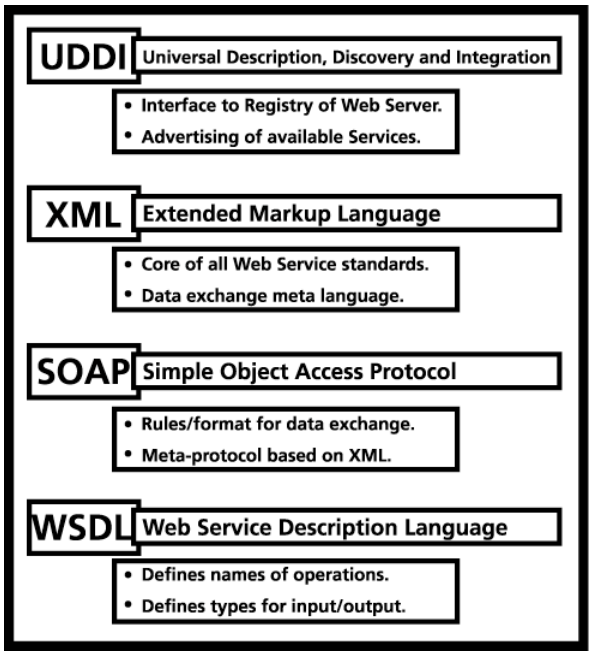


Figure 10: Web-Services environment standards

Exposing the data sources within the data layer and the information-centric components of the information management layer of a knowledge management system to a Web-Services environment provides a means of integrating and conveniently accessing a heterogeneous set of

software applications. By treating these applications as Web-Services and advertising these services in a registry enables the implementation of client applications that can utilize functionality from multiple applications (i.e., Web-Services). Clients can discover services based on service type, rather than being restricted to a specific service at a known location. The use of SOAP and other XML-based languages for communication frees both server and clients from dependence on a particular programming language or operating system.

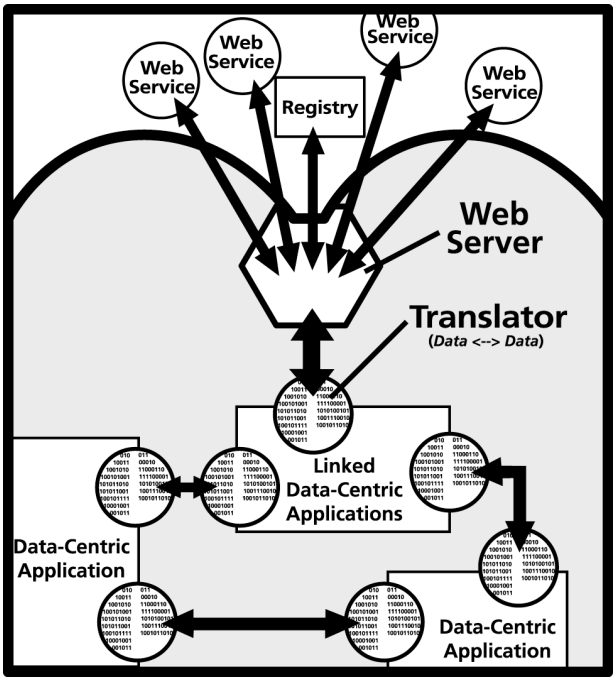


Figure 11: 'Exposing' a data-centric application to a Web Server

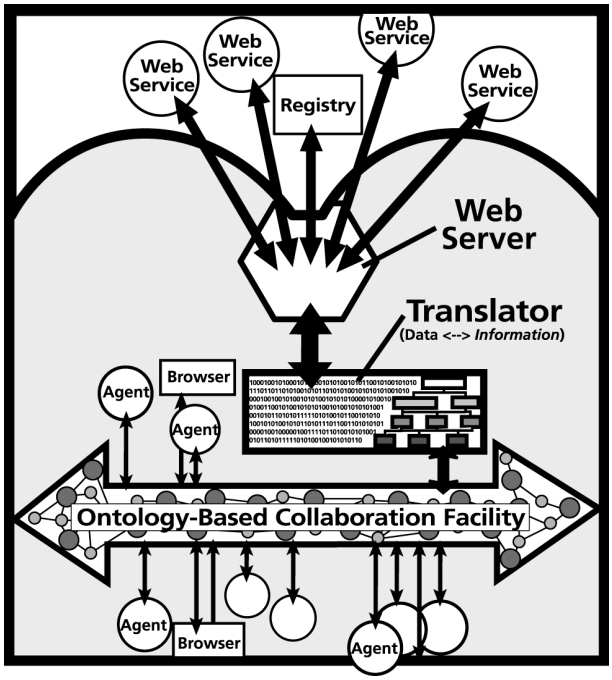


Figure 12: 'Exposing' an information-centric application to a Web Server

2.3 Definition and Explanation of Terms

The purpose of this Section is to provide explanations of a number of terms that are commonly used in conjunction with information technology in general and the DoD net-centric data strategy in particular.

Agents: This term has been applied very loosely in recent years. There are several different kinds of agents. Symbolic reasoning agents are most commonly associated with knowledge management systems. These agents may be described as software modules that are capable of reasoning about events (i.e., changes in data received from external sources or as the result of internal activities) within the context of the information contained in 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.

Since such ***software agents*** are also often referred to as ***intelligent agents*** the question arises whether computer intelligence is really possible or a misnomer? From a commonsense point of view it would appear that humans have intelligence and computers are just very fast but

unintelligent machines. Looking at this question from an entirely human point of view we may well come to such a conclusion. However, are there different kinds of intelligence? In other words, is intelligence something that is entirely reserved for living beings or can a machine display behavior that is akin to intelligence?

Before attempting to answer this question we should perhaps first address another seemingly less difficult question: Are there different levels of intelligence? If there are levels of intelligence then remembering is probably the lowest level of intelligence. Certainly computers can store vast amounts of data and can retrieve these data quickly and accurately. The immediate response might be that remembering is more than just retrieving data. Remembering also involves relationships and context. It is this context that makes data meaningful and relevant. By including an information model (i.e., ontology) in a software application we are able to represent information rather than data in the computer (i.e., information is data with relationships to provide some degree of context). This allows us to include modules in the software (i.e., software agents) that are able to automatically reason and communicate the results of their reasoning activities to other agents (including human users). One could argue that in some respects we are creating in this way a virtual copy of a problem situation, or even a limited form of human society, in the computer environment. The players (i.e., the agents) in this virtual society can assume many different roles and can contribute and collaborate at many levels; - from the most primitive to the more sophisticated levels.

Therefore, if we store not only data in the computer but also the relationships that convert such numbers and words into information then we can also embed in the software rule sequences that are capable of reasoning about this information. Such sequences may be as simple as condition-action statements. For example, ***if*** an enemy tank unit is sighted ***then*** place a call-for-fire on the enemy tank unit ***and*** commence the process of weapon selection. In this way we can implement, through the use of computer software, at least some elementary automatic reasoning capabilities in computers. As an example, a decision-support application for expeditionary warfare involving sea-basing operations might include agents that perform very elementary tasks such as calculating the fuel consumed by a helicopter in transporting supplies from the sea base to an inland supply point.

However, the same application might also incorporate agents that perform more sophisticated tasks. For example, selecting the best mix of lift assets (e.g., helicopters, hovercraft, vertical take-off aircraft, etc.) to transport a wide range of supplies to multiple landing zones within requested time windows, and within constraints such as weather conditions, enemy actions, and so on. The latter agents consider results received from other agents, and utilize a wide range of heuristic and algorithmic methods to arrive at a possible solution. In some respects this is similar to human society where problems are often solved through a team effort. In such teams some people contribute very simple capabilities and others contribute more sophisticated capabilities. Are the combined actions of these agents totally predictable? The answer is, no. While the results produced by the simple agents are certainly predictable, the impact that these results may have on the collective actions of the system is not necessarily predictable. In other words, the intelligence of the software system derives from the interactions (or more appropriately the collaboration) of the communicating elements of the system (i.e., the agents). This is a ***collective intelligence*** that is not necessarily predictable and that can be quite powerful.

COI: Communities of Interest (COIs) are composed of groups that pursue common endeavors. Such groups typically share similar goals, objectives, problems, business processes, and vocabulary. Experience has shown that attempts to standardize nomenclature, logical data models, business rules, and processes at an enterprise level are usually counterproductive and eventually abandoned. Similar attempts to establish and enforce standards within smaller and more focused communities have been more successful, although not entirely without obstacles. Difficulties arise when COIs are not sufficiently specialized or when established standards are not adequately enforced within the community. DoD's Global Information Grid (GIG) vision assumes that with the explicit formal declaration of metadata it will be possible to automate the mapping of data and information that is required to be exchanged between any two COIs.

Data: Strictly speaking data are numbers and words without relationships. Even though data are often stored in a relational database management system, typically only minimal relationships are stored with the data. Without adequate relationships, data do not contain sufficient context to support automatic reasoning capabilities by software agents.

Software that incorporates an internal representation of data (i.e., numbers and words) with few (if any) relationships is often referred to as ***data-centric*** software. Although the data may be represented as objects the scarcity of relationships, and therefore the absence of adequate context, inhibits the inclusion of meaningful and reliable automatic reasoning capabilities. Data-centric software, therefore, must largely rely on predefined solutions to predetermined problems, and has little (if any) scope for adapting to real world problems in near real-time. Communication between data-centric software applications is typically restricted to the passing of data-string messages from one application to the other. This imposes a larger transmission load than communication between information-centric applications. Since a data-centric application has no 'understanding' of the data that it is processing, a complete set of data must be transmitted so that the receiving application can process the transferred data in the appropriate predefined manner. For example, if the data to be transmitted involves the new location of an automobile then a complete set of data describing the automobile (including its new location) must be transmitted. In the case of information-centric applications only the new location and some object identifier would need to be transmitted, because both the transmitting and receiving applications have some 'understanding' of the general notion of an automobile and the specific instance of that notion representing the particular automobile that has changed its location.

Data Mining: Data Mining tools are applications that are designed to analyze the data in a Data Warehouse (or Data Mart) to establish relationships, identify trends, and predict future trends.

Data Portal: Typically, Data Portals are designed to provide access to operational data, with an emphasis on the presentation of data (usually to human users). Data Portals may also incorporate data analysis tools, and are often accessed in a Web-Services (e.g., Internet) environment. A Data Portal typically does not store data but provides a single point of entry to heterogeneous data sources.

Data Warehouse: Data Warehouses most commonly store and manage summarized (i.e., archived) data, usually in a relational database management system. However in some cases, for example when the operational data sources are particularly fragmented and unreliable, a Data

Warehouse may serve as an operational data store that cleans and integrates data at the same granularity as they exist in the original data sources. The summarized or atomic data are periodically updated according to a predefined timeline. Data Warehouses often employ sophisticated data indexing mechanisms (e.g., based on key word indexing schemas) to facilitate the rapid retrieval of data. A subset of the data stored in a Data Warehouse that is focused on a particular functional area, is commonly referred to as a ***Data Mart***.

GES: Within the broad GIG vision, the GIG Enterprise Services (GES) will provide computing services that can be used by COIs 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. The GES 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).

GIG: The Global Information Grid (GIG) is the vision that will guide the DoD in the implementation of an integrated network of knowledge management services and capabilities. Succinctly stated the GIG is 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, 1). The implementation of this vision will require: (1) the standardization of nomenclature and reference tables; (2) the definition of logical data models; (3) the publication of data encoding protocols and formats (i.e., metadata registries); (4) the adoption of data transmission standards; (5) the development of functionally created ontologies that allow data to be placed in context; (6) the publication of business rules and the encoding of these rules in software agents with automated reasoning capabilities; and (7) the formulation of policies, conventions, and processes, designed to facilitate planning, problem solving, and decision-making tasks.

All of these requirements 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. 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 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.

Information: Information refers to the combination of data with relationships to provide adequate context for the interpretation of the data. The richer the relationships the greater the

context (i.e., meaning conveyed by the combination of data with relationships), and the more opportunity for automatic reasoning by software agents.

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

Knowledge: The term knowledge is commonly used to refer to the body of data, relationships, identified patterns, principles, rules, problem solving methods, and prototype solutions that are known in any domain. It may be explicitly recorded and described in text (e.g., books, journals, minutes of meetings, etc.) or implicitly held in the cognitive system of individuals. Knowledge is typically acquired through experimentation, observation, analysis, and experience by human beings. The principal purpose of ***knowledge management*** is to make knowledge explicit and accessible throughout an organization, so that: (1) groups within the organization can effectively collaborate in the performance of tasks; (2) new members of the organization can rapidly acquire the necessary knowledge to become productive; (3) lower level information analysis and reasoning tasks can be automated; and, (4) past decisions, solutions and lessons learned can be captured and considered in future tasks.

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 5, the Metadata Registries and Catalogs would reside in the Mediation Layer while the actual data (i.e.,

Shared Spaces) would be found in the Data Layer. The range of metadata that is planned for the DoD Metadata Registry includes: standard reference tables; XML, EDI, X-12, EBXML, and similar formats; logical data model structures; symbologies; taxonomies; ontologies; and transformation services.

OLAP: On Line Analytical Processing (OLAP) tools are applications that are typically utilized to extract answers to Who?, What?, and Why? queries, constrained by the need for users to understand the structure of the database since the contents of the database are not represented in terms of the user's domain.

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

3. The Existing USTRANSCOM Corporate Data Environment (CDE)

It is not the intent of this Section to discuss at any length the systems, operational and technical views of the USTRANSCOM Enterprise Architecture. Detailed descriptions of these views are provided in the Defense Transportation System Enterprise Architecture (31 January 2003) document. Instead, the purpose of this Section is to provide a brief summary of the principal components of the architecture and the principles that have been established by the Corporate Data Office (CDO) to discipline the data environment and prepare it to become part of the foundation of a knowledge management environment.

The commencement of formal work on USTRANSCOM's CDE dates back to March 1991 with the publication of USTRANSCOM Regulation 4-5. This led to the development of the Transportation Logical Data Model (TLDM) and a strong commitment to the OSD effort aimed at instituting a DoD-wide data administration program based on DoD Directive 8320. The TLDM is now referred to as the Master Model (MM) by USTRANSCOM.

Largely due to the sterling efforts of its Corporate Data Office (TCJ6-AD), USTRANSCOM has established the foundations of a well defined, documented and disciplined data architecture that is consistent with DoD data management policies (USTRANSCOM 2003b) and favorably positioned to embrace and exploit DoD's rapidly evolving plans for implementing a Global Information Grid (GIG) as a net-centric environment of seamlessly interconnected data sources and utilization capabilities (DoD Net-Centric Data Strategy, Stenbit, May 2003).

3.1 Overview of the CDE Architecture

As it currently exists, USTRANSCOM's CDE is largely confined to the data-centric aspects of the full set of capabilities that a knowledge management environment would be expected to provide. It essentially consists of three groups of components: a data layer that addresses the storage and integration of data contributed by heterogeneous data sources; a number of repositories and registries containing architectural descriptions, business rules, and metadata definitions; and, a proliferation of internal and external legacy applications that were designed to facilitate the acquisition, processing and display of data (Figure 13). The ability of the Corporate Data Office to ensure the compliance of these applications with the Master Model has been somewhat compromised by the fact that a significant number of the legacy applications had existing databases prior to the establishment of the TLDM, and that USTRANSCOM did not have full control over some of the new applications (e.g., the original GTN project).

During the past several decades, like virtually all government agencies and larger commercial organizations, USTRANSCOM has increasingly struggled to maintain control of a burgeoning data load. As the need for responsiveness in terms of faster and more accurate transportation planning and replanning capabilities increased, USTRANSCOM and its component commands added a variety of analysis and planning-assistance tools to their individual data-processing environments. Some of these applications (e.g., DAAS, JOPES) came from external sources with relatively little USTRANSCOM control and influence, while others (e.g., CAMS GO81, ATMS, WPS, IBS) were initiated, designed, and implemented under internal control. As was commonly the case in the pre-1990s period, each of these applications was purposely designed to be process-oriented and consequently considered data to be a private rather than a shared resource. This contributed greatly to a stove-piped data-processing environment in which any attempt to

link applications horizontally required the development of complex interfaces capable of transforming the data schema and format of one application to that of the other. The development of such data bridges was an expensive undertaking, and once a linkage between two stove-piped applications had been accomplished it constituted a potential failure point and required continuous maintenance.

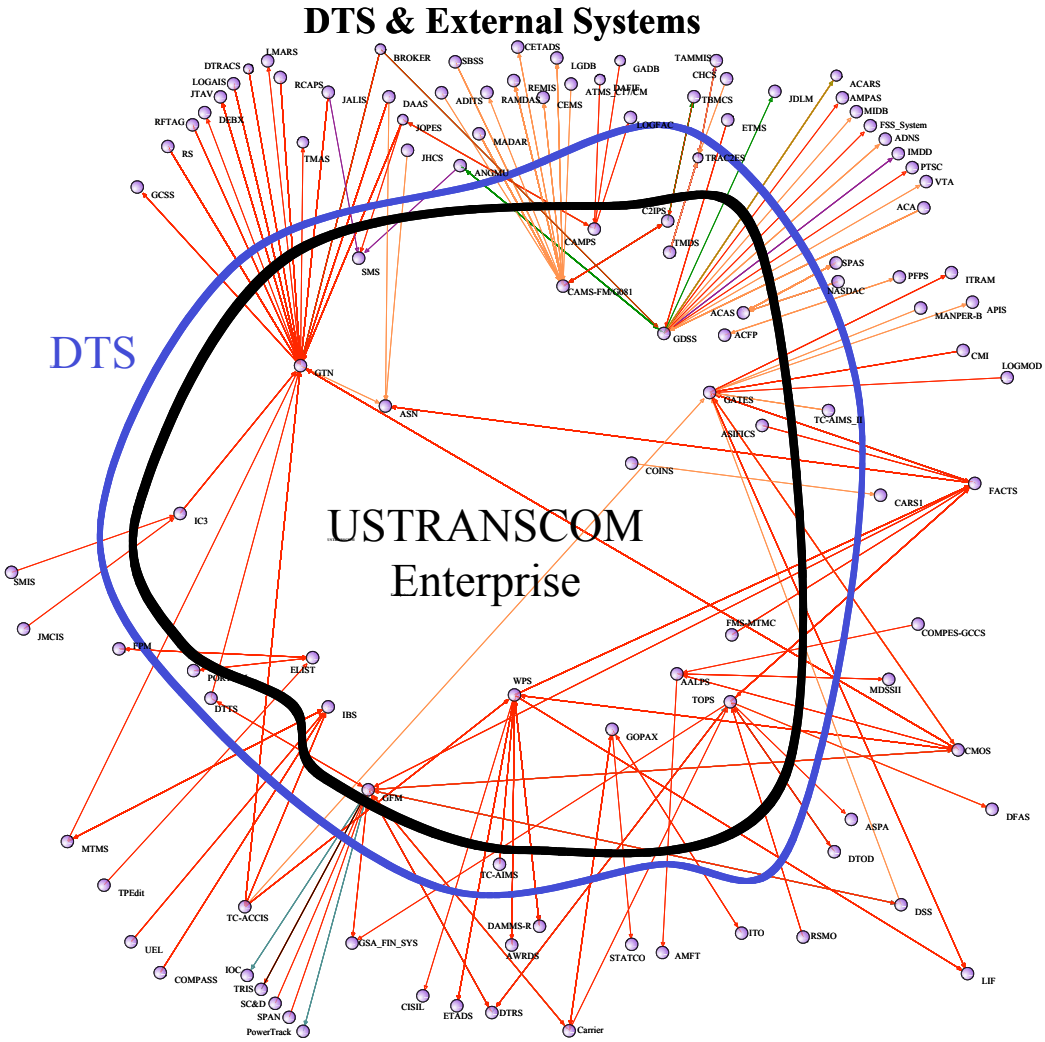


Figure 13: USTRANSCOM internal and external system interfaces (October 2002)
(Source: Figures TV-1-2, DTS Enterprise Architecture, 31 January 2003)

Over time the proliferation of applications created a degree of functional overlap, lack of interoperability, and inordinate maintenance burden that could no longer be tolerated by USTRANSCOM. Accordingly, in the 1990s USTRANSCOM under direction of OSD initiated through its Joint Transportation Corporate Information Management Center (JTCC) a systematic review and evaluation of all of its computer-based transportation systems. Specifically, OSD sought the elimination of functional duplication in multiple systems and the migration of service

unique applications to joint transportation systems. In July 1995, the Deputy Under Secretary of Defense (Logistics) approved 19 ***migration systems*** for the Transportation functional area.

Concurrently, recognizing the increasingly urgent need for achieving a useful level of interoperability among existing and new applications, USTRANSCOM embarked on a concerted effort to integrate and discipline its data environment. Initially, utilizing the IDEF (Integrated Definition for Function Modeling) family of methods, this effort focused on the development of a data dictionary, a logical data model, and the establishment of principles, policies and processes in support of enterprise integration. Today, the products of these initial efforts are maintained by the Corporate Data Office and represent the most useful existing components of the CDE capable of serving as foundational units of the proposed USTRANSCOM CICE.

The strategies used by the CDO to establish order and discipline in the CDE are clearly reflected in the architectural diagram shown in Figure 14. Faced with the reality that USTRANSCOM has little (if any) control over the more than 80 external government and commercial systems, and only limited control over the 25 or more systems used by its three component commands, the CDO focused its efforts on the creation of a CDE layer that would provide a repository of fully defined, integrated and cleansed operational data within the USTRANSCOM operational domain.. This layer is currently referred to as the Corporate Data Solution (CDS).

The following concepts, which appear to have been employed in the design of this solution (i.e., CDS), are logical and based on sound information technology and software engineering practices:

Standardized Nomenclature: Agreement on a common vocabulary that can be defined within a data dictionary and become part of the DoD Data Dictionary System (DDDS).

Logical Data Model: Development of a single integrated logical data model that is compliant with DoD standards and implemented in third-normal form in adherence with standard relational database management practices. The Transportation Logical Data Model, later renamed as the Master Model, is the product of this effort.

Standard Reference Tables: Identification of data elements that are, or should be, invariable across the entire Enterprise (e.g., National Stock Numbers, hazardous material type identifiers, military vehicle and equipment types and characteristics, symbols, etc.). These data values must not be changed by processes or applications, and should therefore be centrally controlled and maintained.

Policies and Processes: Establishment and enforcement (to the extent feasible within economical, technical and political constraints) of principles and policies, as well as periodic evaluation and compliance processes, designed to facilitate the movement toward a single, well structured, fully defined CDE data framework. These principles, policies and processes are described in the USTRANSCOM Data Management Handbook, August 2003.

Operational Data Stores: Implementation of one or more operational data stores in which the data pertaining to a particular community of interest has been subjected to a formal validation and cleansing process. The Extract-Transform-Load (ETL) process adopted by CDO is applied at the time when data are first brought into the Operational Data Store from a data source.

Metadata Registries: Capture of both data schemas, exchange formats, and transformation rules that describe the structure of data models, repositories, interfaces, mappings, and extraction conventions, as well as business rules that are used in the analysis and exploitation of the actual data. This includes oversight of the Transportation Namespace in the DoD XML Registry in coordination with the Functional Transportation Namespace Manager.

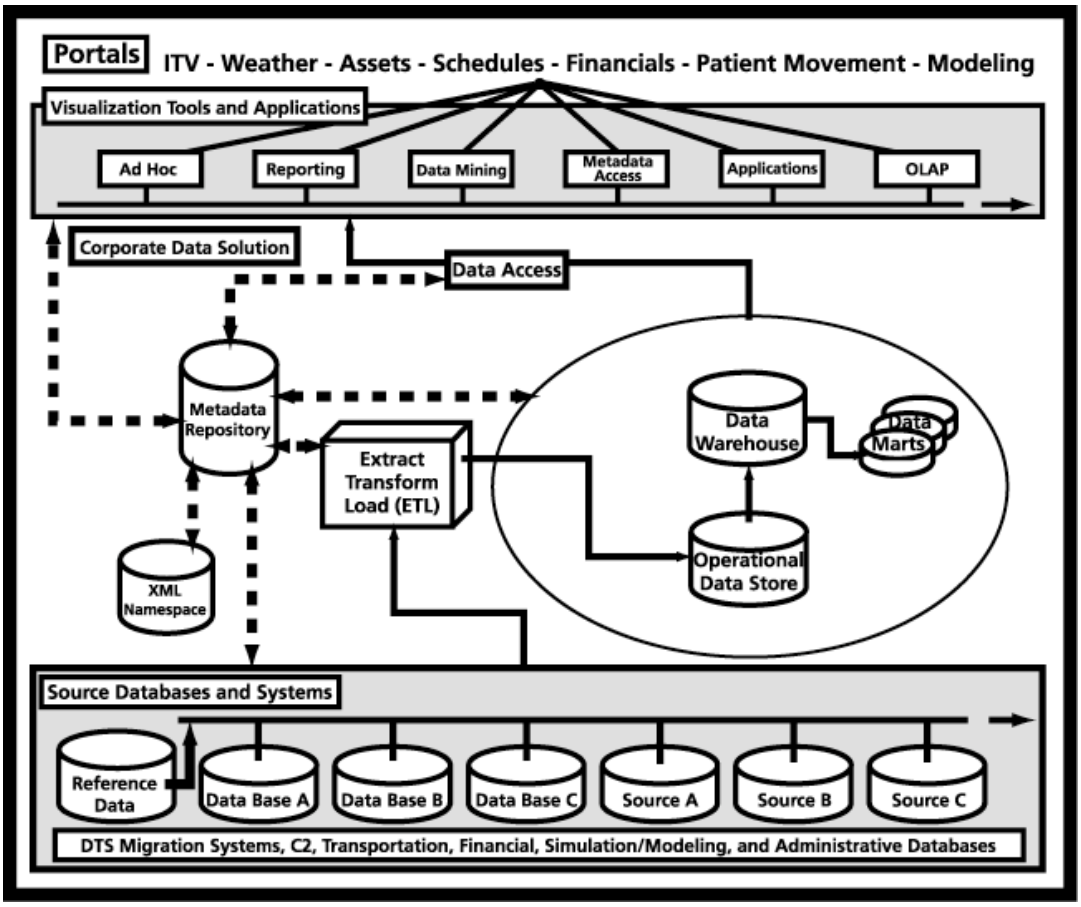


Figure 14: The existing Corporate Data Environment (CDE)

As was probably expected at the outset, the actual implementation of these strategies was no doubt met by many obstacles ranging from lack of funding, inability or unwillingness of project management offices to assign a high priority to the compliance of new systems to the established standards, the existence of a large number of legacy systems that pre-dated the Master Model, and the process-oriented needs and objectives of the users of the data. Nevertheless, the current CDE incorporates significant elements of both the organizational framework and physical components that are necessary as the foundation of a multi-layered information-centric knowledge management environment (see Section 1.1).

The proposed USTRANSCOM CICE described in Section 4, will require the addition of a Mediation Layer and an Information Layer. The only major structural change that will be required in the existing architecture of the CDE (Figure 14) is the transfer of the metadata, ETL

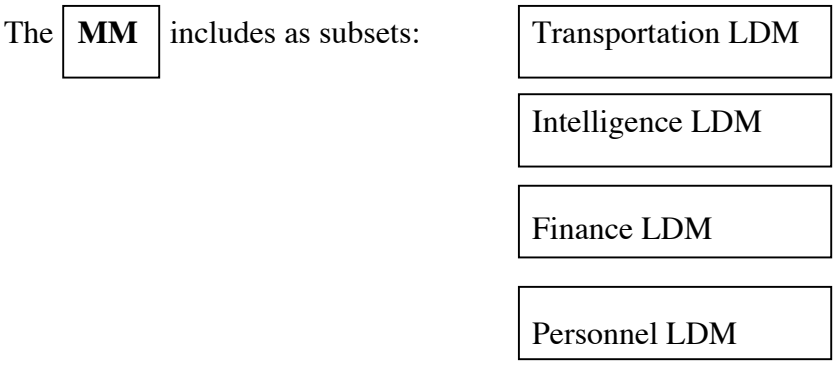
and business rules repositories from the Data Layer to the Mediation Layer. Together with the Corporate Resource Information Source (CRIS) these two existing components can define the beginnings of a Mediation Layer that will allow USTRANSCOM to rapidly proceed with the incremental construction of an Information Layer of ontology-based applications incorporating intelligent, collaborative, decision-assistance tools (i.e., software agents).

3.2 CDE Implementation Principles

USTRANSCOM's Corporate Data Office (CDO) overseas the development and implementation of the policies and procedures that govern the CDE. It executes this responsibility within the framework of the following principles:

Principle (1) – Corporate Logical Data Model (LDM): The Master Model is a single integrated Logical Data Model (LDM), which serves as the foundation for all other USTRANSCOM databases, interfaces and applications. Specifically, the CDO has defined the following rules to ensure adherence to this principle:

- Data must be separated from the application.
- Data need to be managed as a corporate (i.e., enterprise) asset.
- Logical Data Models have proprietary control over applications.
- Maintenance of the structure of the data should be separated from the use of data.
- A single integrated LDM (i.e., the Master Model) is the blueprint for all USTRANSCOM databases and interfaces.



Within the context of the DoD Net-Centric Data Strategy (Stenbit 2003) USTRANSCOM considers itself to be a single Community of Interest (COI) with one LDM, namely the Master Model. It is therefore planned that all USTRANSCOM interest areas will eventually be governed by the Master Model. Currently the Master Model covers: (1) transportation; (2) corporate information resources, including architectures and strategic plans (in CRIS); and, (3) personnel. Future additions are likely to include: (4) finance; (5) intelligence; (6) aircraft maintenance (AMC); (7) contracts (MTMC); and, (8) inspections such as customs inspections (i.e., MTMC and AMC).

Each USTRANSCOM Component Command (TCC) is expected to have its own LDM as a subset of the MM, and each TCC application is expected to comply with the LDM of its home TCC.

It is of interest to note that DoD has recently concluded that a single DoD-wide LDM is not a feasible proposition and, therefore, in January 2003 rescinded its plans for a DoD-wide LDM (i.e., DoD 8320.1) as part of the Defense Data Dictionary System. Instead, DoD appears to be willing to rely on intelligent ***mediation*** to achieve interoperability among the LDM-based databases of the various COIs.

Principle (2) – Sound Engineering Process: USTRANSCOM recognizes that the vital goal of interoperability cannot be achieved if the Master Model is not implemented throughout the CDE. Accordingly, all new applications and systems are required to comply with DoD data standards (including the Master Model) in a physical database. For existing systems, where the necessary reengineering effort may not be feasible due to financial and or other constraints, a system interface approach is permitted. This requires the development of a software interface that transforms the data into a form that is compliant with the Master Model.

Principle (3) – Effective Monitoring Mechanism: An annual Technical Assessment process has been implemented to evaluate the status of each USTRANSCOM system in respect to its compliance with enforced data standards, regulations, and policies. This process is seen as an effective vehicle for encouraging Project Management Offices to progressively implement DoD data standards in their database designs and interface software.

Principle (4) – Single Conduit for Source Data: The Corporate Data Solution serves as the integrator of the CDE and ensures that all source data has a single entry conduit from the heterogeneous data sources into the integrated data environment. The CDS is considered to be a layer within the current CDE architecture, in support of decision-making (Figure 15). Integration is accomplished through a single set of documented ETL business rules. These ETL rules are used to cleanse the data as they are transmitted from the data sources into the Operational Data Store (ODS). The ETL process occurs only once, whenever data are brought from a data source into an ODS. In fact, however, different applications will send the same data items through the appropriate ETL rules to the ODS. How the ETL facilities should deal with the likely occurrence of data conflicts under these conditions is a potentially complex problem that is being addressed by the GTN project.

3.3 Components of the CDE Architecture

Operational Data Store: The CDE architecture incorporates the notion of an ODS. An ODS is typically oriented toward a major functional area (i.e., an interest area such as operations, finance, personnel, etc.). The data in the ODS are extracted through the ETL process from the heterogeneous data sources (Figure 15). In this manner the ODS is expected to provide an integrated corporate view.

The data flow is essentially ***one-way*** from the data sources to the ODS. Changes in data initiated at higher decision-making levels are transmitted back to the lower functional levels by telephone

and similar communication, where they are re-entered as new or changed data into the bottom level data sources. It should be noted that applications that access or generate these data sources at the bottom level will typically exchange data among themselves, instead of receiving their data from the ODS.

It appears that GTN-21 is currently the only program within USTRANSCOM that is constructing an ODS that meets all criteria. However, the GTN-21 contract is still in the design phase. Its predecessor, the GTN program with GTN-Build5 as the final product, falls far short of the established CDE principles.

- Does not meet LDM criteria.
- Does not meet DoD standards.
- GTN-Build5 meets the needs of the GTN application (i.e., it is process-centric). Its data can be accessed only through GTN interfaces or through special SQL queries implemented by database managers.
- GTN therefore serves only its own data needs, and not the data needs of the enterprise (i.e., it imports its own data and does not contribute data to the CDS).

GTN has designed a LDM and a set of ETL rules, but these have not been implemented in a physical database. Clearly, USTRANSCOM does not have full control over all systems (e.g., OSD funds and controls GTN-21). This is potentially counterproductive and essentially forces USTRANSCOM to build an integrated CDE for its own systems only, and then try to interface with the other external systems as best as it can.

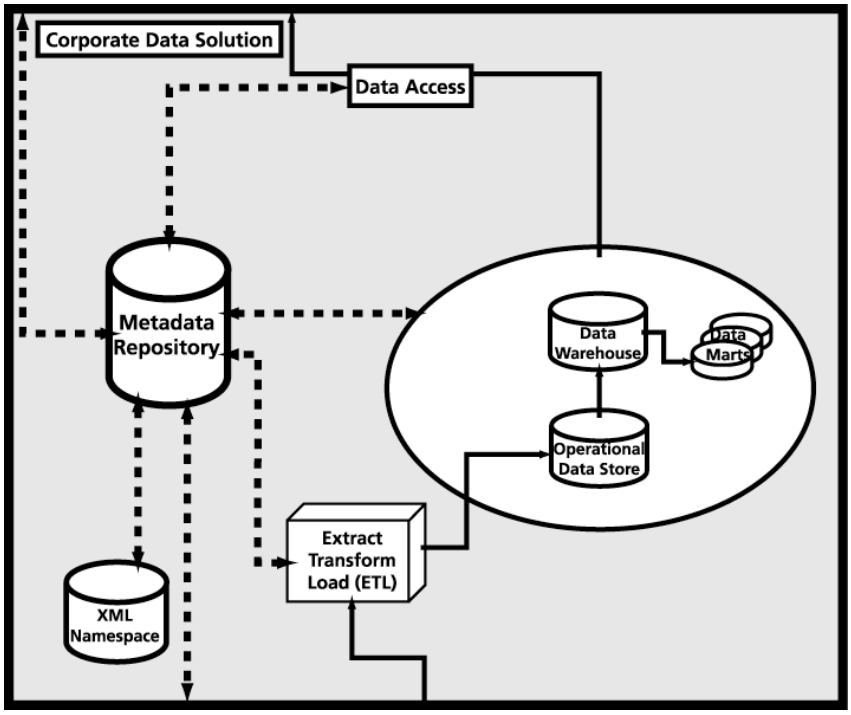


Figure 15: The Corporate Data Solution (CDS) layer of the CDE

Data Warehouse: Within the context of the existing CDE, USTRANSCOM sees the need for Data Warehouses with specialized Data Marts to take advantage of commercially available data analysis tools (i.e., OLAP and Data Mining). This is understandable because only data-centric capabilities are currently available in support of the planning, problem solving and decision-making processes that are undertaken by the users of the data. The need for such tools may significantly diminish in the future with the availability of much more powerful ontology-based decision-support applications incorporating intelligent software agents. However, Data Warehouses can certainly exist harmoniously within the Data Layer of a multi-layered knowledge management environment (see Section 2.2 (*The architecture of a knowledge management system*) and Figure 5).

Data Warehouses are oriented toward specific subject areas. They contain cleansed and integrated data drawn from source data systems. Two Data Warehouse design strategies are prevalently used.

Design A: Summarizes data within defined dimensions of interest (i.e., aggregates data historically). Typically incorporates 'fact tables' and 'dimension tables' in a star schema. The stars are often referred to as Data Marts.

- ***Advantages:*** economy of storage
simplified queries
- ***Disadvantages:*** difficult to work outside the predefined dimensions of interest

Design B: Stores all (atomic) data in 'normalized' form (i.e., data are not aggregated).

- ***Advantages:*** not constrained by predefined dimensions of interest
- ***Disadvantages:*** increased storage requirements
relatively more complex queries

The BDSS (Business Decision Support System) project incorporates a Data Warehouse, which stores atomic data (per the 'Design B' strategy).

Metadata Registries: Metadata provides information about the location and structure of data, methods for accessing data (whether in data tables or electronically stored documents), the architecture of data environments, rules for transforming data, business rules for processing data, and any other information that can assist in the convenient (automated as far as possible) retrieval and discovery of data. Apart from establishing within the CDE various repositories for metadata, USTRANSCOM has also implemented the CRIS database with the objective of providing a single, reliable source for DTS metadata information. As described in the DTS Enterprise Architecture documents (USTRANSCOM 2003a), the CRIS database already includes a variety of information ranging from system locations, interface definitions, data exchange requirements, operational facilities, organizations, standards, business processes, to strategic planning initiatives, and can be accessed through a Web site interface.

Reference Data: The CDO has implemented a reference table management and synchronization process to ensure adherence to a common set of data values that should not be changed during the processing of data. The Defense Information Systems Agency (DISA) serves as the executive

agent for the centralized management, identification of authoritative sources, and distribution of reference table domain values.

Data Sources: The layer of 'Source Databases and Systems' at the bottom of the CDE architecture diagram in Figure 14 consists of the 22 transportation ***migration systems*** authorized by OSD in 1995, the remaining legacy applications that have not been replaced by the migration systems, other systems owned by the three component commands, and the plethora of external government and commercial data sources that USTRANSCOM must interface with in support of its functional needs (Figure 13).

Portals: The proposed CDE architecture shown in Figure 14 includes a layer of user-interfacing data visualization and analytical tools, together with a set of domain-specific applications. Collectively, the components of this layer are referred to as 'portals'. In a data-centric environment, portals require the users to determine which components match their needs. The result is frequently a combination of missed opportunities (i.e., useful information that the user does not see because it is available through a component that the user has not selected) and irrelevant presentations (i.e., information that is not germane, but is presented to the user as part of a selected component). Information-centric systems can mitigate both of these drawbacks through the ability of the system to reason about the utility of information to a particular user, regardless of the source. Based on models of the components, the underlying domain, and the user's preferences, agents can improve the quality, completeness, and timeliness of the information presented through the portals.

3.4 Users of the CDE

The current users of the CDE include both internal and external parties, with different data and information needs undertaking widely ranging tasks for quite different purposes. Ideally, the views and tools that are required in support of these users should be provided as a small number of flexible, customizable and adaptable applications, rather than a large number of specialized solution-oriented applications. Unfortunately, this far more economical approach is not available in a data-centric environment where the lack of information context requires applications to include hard-coded solutions rather than more generic tools with automated reasoning capabilities.

1.1 Data contributors from outside USTRANSCOM.

1.2 Data contributors from inside USTRANSCOM.

1.3 Information users (i.e., trackers and expediters; - current operations in-transit visibility).

1.4 Information users (i.e., analyzers and planners).

1.5 Decision makers (there are essentially two kinds of decisions that are required to be supported: (1) time critical current operations decisions; and, (2) deliberate planning decisions that deal with longer term issues requiring historical data). The levels of decision-making include the following:

- Commanding General (CG)

- HQ TC staff (e.g., Joint Mobility Operations Center (JMOC))
- Centralized decision making by component commands.
- Distributed decision making by component commands (e.g., at ports).
- Installation Transportation Officers (ITOs) throughout the world.

1.6 Decision makers (i.e., higher level commanders).

1.7 Data and metadata administrators.

1.8 System administrators.

4. Extending the CDE to a Knowledge Management Environment

It has been recognized by DoD and the Department of Homeland Security that the proposed new ***Infostructure*** will require transition from a strictly data-oriented IT environment to an information-centric knowledge management environment. Accordingly, ***it is not a matter of whether USTRANSCOM should extend its CDE to a knowledge management environment, but how quickly this inevitable transition can be accomplished*** and what steps should be taken immediately to facilitate and accelerate the necessary technical and organizational changes.

However, the necessary transition steps are not only technical in nature. They require the human users of the significantly more powerful technical capabilities of a knowledge management environment to reevaluate the very essence of their relationship with computer-based systems. To date the human users have shown a great deal of resistance to the notion that a computer is any more than a dumb electronic machine. The concepts of computer intelligence and human-computer collaboration are not generally compatible with our human view of the world (Pohl 2003). Yet, such computer-based capabilities have been demonstrated as being feasible, have been available for exploitation for more than a decade, and offer the most immediate solution approach for refocusing scarce and expensive human resources from the tedious tasks of managing and interpreting an overwhelming volume of data to the more fruitful tasks of exploiting useful data within the appropriate information context.

It is important to recognize that the human aspects of the transition require as much attention, planning, and nurturing as the technical aspects of extending the existing CDE to a vastly more powerful CICE. Experience has shown that the principal vehicles for achieving an expeditious, effective and relatively smooth organizational transition under conditions of accelerated change are: leadership and vision; multi-level coordination and planning; uninhibited communication; explanation and education; and, a continuous stream of useful incremental results and products (Helgesen 1995, Drucker 1993, Covey 1989).

4.1 Setting the Stage for CICE

There are several converging forces that will continue to exert pressure on USTRANSCOM to extend its CDE and Enterprise Architecture into a more powerful information-centric environment. These are related to economics, diminishing human resources, responsiveness, expanding responsibilities, and information technology advances.

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 a data-processing environment that has hundreds of potential failure points. Such an environment typically has no 'graceful degradation' capabilities.

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 budget is spent on preventing or recovering from crashes" (Patterson et al. 2002). Simply stated,

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.

The implementation of the GIG vision 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 grid could easily exceed this increase by several orders of magnitude. Such a volume of raw data is likely to choke the GIG 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 GIG 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 products generated by the lower level tasks.

As discussed previously in Section 1.2, the changing military context is placing an unprecedented emphasis on responsiveness, interoperability, adaptability, and accuracy. As clearly stated in the Systems View (SV-1.2) and inferred in many other sections throughout the Defense Transportation System Enterprise Architecture document (USTRANSCOM 2003a), the existing CDE is "… incapable of adequately resolving ad hoc tasking" and unable to effectively support interactions among functional areas "… without significant human intervention". The burden currently placed on human users who are required to interpret, filter, and in many cases transform data that could be automatically manipulated if they were represented in a more appropriate form (i.e., within an information model that places the data into context), is unnecessary and unacceptable.

USTRANSCOM, like most large organizations, is currently forced to dedicate a significant portion of its operating budget, staff, project budgets, and time, on the piecemeal resolution of ad hoc problems and obstacles that are symptoms of an overloaded data-centric environment. Examples include: data bottlenecks and transmission delays resulting in aged data; temporary breakdown of data exchange interfaces; inability to quickly find critical data within a large distributed network of data-processing nodes; inability to interpret and analyze data within time constraints; and, determining the accuracy of the data that are readily available. This places the organization in a ***reactive*** mode, and forces the organization to expend many of its resources on solving the symptoms rather than the core problem. In other words, the ability of a knowledge management environment to support: (1) the automatic filtering of data by placing data into an information context; (2) the automated reasoning of software agents as they monitor events and assist human planners and problem solvers in an intelligent collaborative decision-making environment; and, (3) autonomic computing capabilities; - provides a core solution that will eliminate many (not all) of the symptoms that currently demand the continuous attention and engagement of the organization's personnel.

On the positive side, USTRANSCOM is well positioned to undertake the necessary steps to effectively extend its data environment into an information-centric knowledge management environment. The work that has been undertaken with commendable foresight over the past decade to construct a logical data model (i.e., the Master Model), establish a disciplined corporate data environment (i.e., the CDE) with policies, principles and processes, implement and enforce these to the extent possible, and stay closely aligned with OSD's vision of a global information grid (i.e., the GIG), will allow USTRANSCOM to proceed advantageously with the implementation of CICE.

4.2 Proposed CICE Architecture

Even though the existing CDE has a decidedly data-centric orientation and is therefore not structured as a knowledge management facility, it already includes several major components that are fundamental to a knowledge management environment. Among these are the Master Model, the Reference Data Tables, the Metadata Repository, and the CRIS Repository. In addition, all of the efforts that have been directed toward the cleansing and integration of data resulting in the creation of an ODS and a Data Warehouse with Data Marts will greatly contribute to an early realization of the benefits of an information-centric reorientation.

The architecture of CICE, shown in Figure 16, is conveniently divided into four layers. The complexity of the many external interfaces (Figure 13) suggests that the single data layer that is common in knowledge management systems should be subdivided into a Data Source Layer and a Data Integration Layer. However, apart from this adjustment the proposed CICE architecture follows the established principles of a knowledge management system environment described in Section 2.2 and Figure 5. The Mediation Layer provides the necessary interface between the Data Integration Layer and the Information Exploitation Layer.

Contrary to the current one-way operation of the CDE, the flow of data in Figure 16 is shown to flow upward and downward. The upward flow proceeds from the data sources through the Data Integration Layer where the data are cleansed to the extent possible and integrated within the ODS and/or the Data Warehouse. Metadata in the Mediation Layer, such as ETL rules and interface formats, are typically used in this process. From the Data Integration Layer (i.e., ODS and Data Warehouse) only those data items that have been required and are relevant within the context of the ontology-based applications residing within the Information Exploitation Layer are mapped to the information models of these applications. Again, the Mediation Layer provides the required translation and mapping rules that are required in support of this process. The CICE architecture shown in Figure 16 foresees a two-way flow, with the results of the planning and decision-making processes that occur within the Information Layer being pushed down to the Data Source Layer.

The one-way upward flow of data from the data sources to the ODS or Data Warehouse appears to be due to the current scope of the CDE rather than any functional or operational requirements. It places what would seem to be an unnecessary burden on the human operators to pass the results of decisions down to lower levels by other communication means (e.g., telephone, e-mail) where the data changes are re-entered into the CDE. More appropriately any changes in data that occur directly at the time decisions are made should be immediately recorded so that they are available to any other planning and decision-making processes. This is a prerequisite if the knowledge management environment is required to achieve a 'zero latency' state (i.e., the

removal of latency from operations so that business events that occur anywhere in the enterprise can immediately trigger appropriate actions across all other parts of the enterprise).

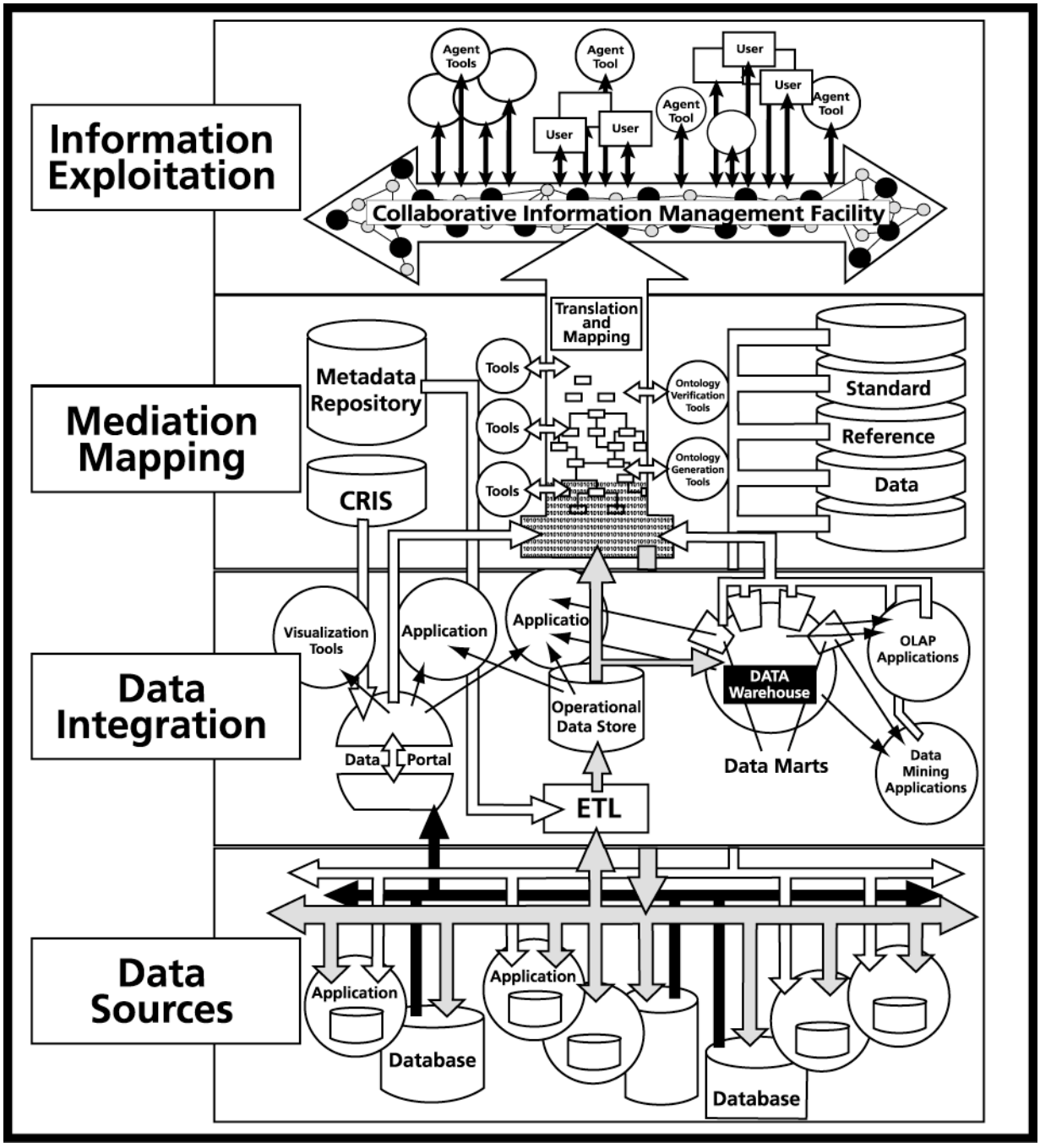


Figure 16: USTRANSCOM Corporate Information-Centric Environment (CICE) architecture

There are no aspects of the CICE architecture that would prevent support of a two-way flow of data-to-information-to-data, if the underlying Data Layer provides the required support. However, the implementation of this capability has at least three implications at the data and

information levels. First, it will require CICE to keep track of the source of data items as they are mapped to the information model of any particular ontology-based application. Second, the Mediation Layer will be required to resolve any potential conflicts that may arise when attempts are made to concurrently update the same data items with different values from several applications. The resolution of such parallel actions will require the time stamping of data and the assignment of prioritization rules. Third, each of the applications that caused the data update action needs to be notified of the final value assigned to a deconflicted data item in case the resolved value requires further action by any one or more of these applications.

While this certainly increases the implementation complexity of CICE (however, only in the case of data originating from the Data Source and Data Integration Layers), it is nevertheless a technical issue that should be transparent to the users of the knowledge management environment. Also, the technical implementation requirements are well within the scope of current software engineering and development capabilities. For example, the need to notify an ontology-based application that a data value that it has just operated on has again changed due to some external action is an automatic consequence of a subscription service. Based on the subscription profile of the application, such a service will automatically notify all clients of any information changes that have occurred within the realm of their subscription profiles.

It should be noted that the achievement of 'zero latency' is an idealized objective that depends on several factors besides the support of two-way data flow. In other words, the goal of 'zero latency' may be theoretically feasible but difficult if not impossible to realize in practice. Among the determining factors are communication network performance limitations in respect to need, bandwidth, latency and throughput capacity, as well as computer hardware and software restrictions. While near 'zero latency' may be achievable within a single distributed application, it is unlikely to be achieved throughout a larger knowledge management environment in which many distributed systems interact over multiple networks.

Data Source Layer: At the bottom level of the CICE architecture are the migration systems and applications, USTRANSCOM and TCC databases, other legacy systems, and sundry external data feeds. These data sources constitute the principal driver of the knowledge management environment. Any new or changed data are automatically passed on to the Data Integration Layer.

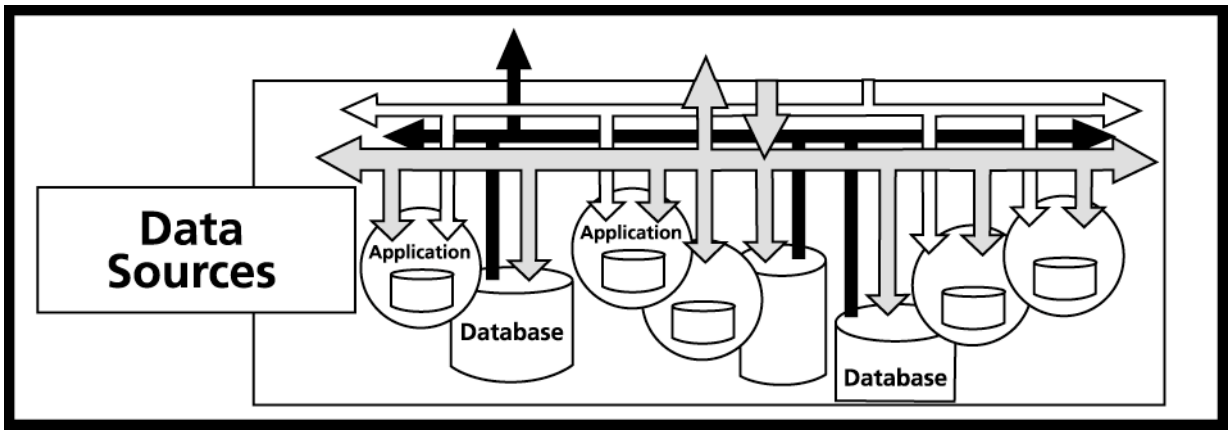


Figure 16a: The Data Source Layer of CICE

Two other aspects of the Data Source Layer require further discussion. First, as shown in Figure 16a, most (if not all) of the databases in this layer should be accessible through one or more Data Portals that are themselves resident in the Data Integration Layer. This is compatible with the normal role of a Data Portal as a gateway to heterogeneous data sources and a vehicle for facilitating the visualization of data by human users. Second, the Standard Reference Tables belong in the Mediation Layer but should be accessible by any of the applications resident in the Data Source Layer. These tables are maintained by database management staff and are by their nature more closely associated with metadata than data sources. However, as shown by the downward arrow emanating from the Mediation Layer, they need to be accessible to the applications that are resident in the Data Source Layer.

Data Integration Layer: This is the appropriate layer for housing the ODS(s), Data Warehouse(s), Data Marts, and Data Portals (Figure 16b). Also included in this layer are any data analysis tools and applications that operate directly on ODS and Data Warehouse data (e.g., OLAP and Data Mining applications), as well as visualization tools that are associated with Data Portals.

The ETL rules that are used to cleanse and integrate data as they are uploaded from the various data sources into one or more data stores (e.g., ODS) are applied in the Data Integration Layer, but stored as metadata in the Mediation Layer.

It should be noted that all of the existing data-centric applications, including most if not all of the migration systems (ICODES as an information-centric application might be an exception), the data analyses applications, and the data visualization tools, are appropriately located in the two bottom layers (i.e., the data layer of the knowledge management environment).

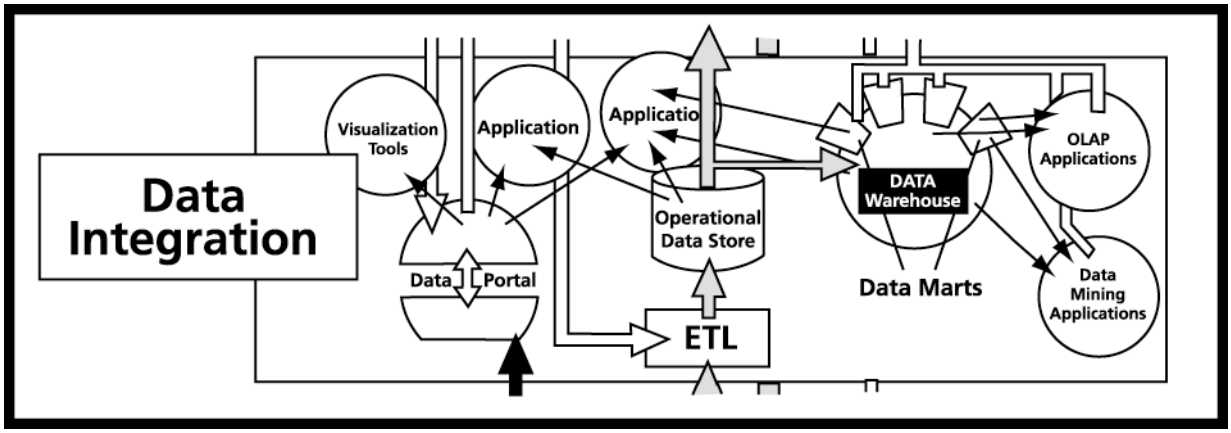


Figure 16b: The Data Integration Layer of CICE

Mediation Layer: This layer performs two important roles within the CICE architecture (Figure 16c). First, it serves as a repository for all descriptions that are required to define data schemas and data exchange formats, corporate resource information, the Master Model, the Standard Reference Tables, ontologies, symbology, business rules, ETL rules, and any other specifications that fall into the category of metadata. Although some of these metadata repositories and operational data stores are pictured separately in Figure 16c (for the sake of clarity), they may well be stored in the same database system. Second, the Mediation Layer is responsible for

interfacing between the Data Integration Layer and Information Exploitation Layer. This involves: (1) mapping of data from the ODS, Data Warehouse and Data Marts to the various ontology-based applications; (2) translation of information changes that occur within ontologies into the equivalent data forms, thereby triggering the updating of the appropriate data stores and sources in a two-way data flow (as shown in Figure 16c); (3) verification of the compliance of individual ontologies with the Master Model; and, (4) generation of ontology templates from the Master Model to facilitate the ontology development process.

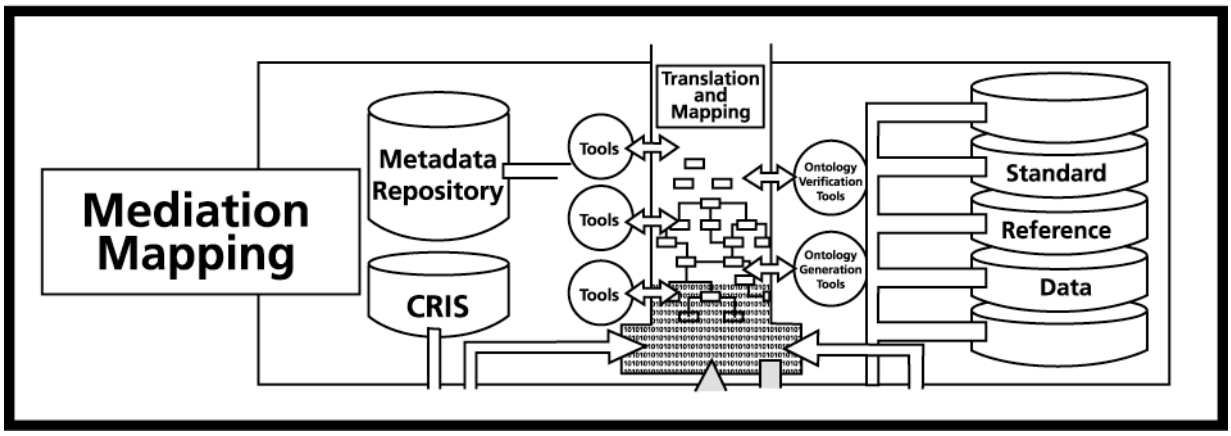


Figure 16c: The Mediation Layer of CICE

In this second role the Mediation Layer will provide a progressively more comprehensive set of tools, which are referred to in the DoD Net-Centric Data Strategy (Stenbit 2003) as GIG Enterprise Services (GES). In fact, many of these tools are likely to be provided by DISA as the executive agent responsible for the development and maintenance of GES. In some cases, such as tools required for the verification of Master Model compliance and ontology template generation, USTRANSCOM may decide to proceed with development rather than delay the implementation of CICE while DISA is fully engaged with other priorities.

Information Layer: As schematically depicted in Figure 16d, this layer is essentially the information management engine of CICE. Fundamental to its intelligent and collaborative assistance capabilities is the representation of information, rich in relationships, within ontology-based applications. As mentioned previously (see Sections 1.1 and 2.3), ontologies are functionally oriented with a focus on the exploitation of data. A logical data model such as the Master Model, on the other hand, provides a mathematically verifiable structural representation of the data entities that apply to a particular domain. It follows that there exists a one to many relationship between a logical data model and the ontologies that facilitate the utilization of data within the often complex relationships (i.e., context) that govern the planning and decision-making activities within a functional area. In this respect an ontology adheres to the nomenclature and fundamental entity relationships that are defined in the logical data model, and builds on top of these the additional relationships and constraints that describe the context in which the data will be utilized.

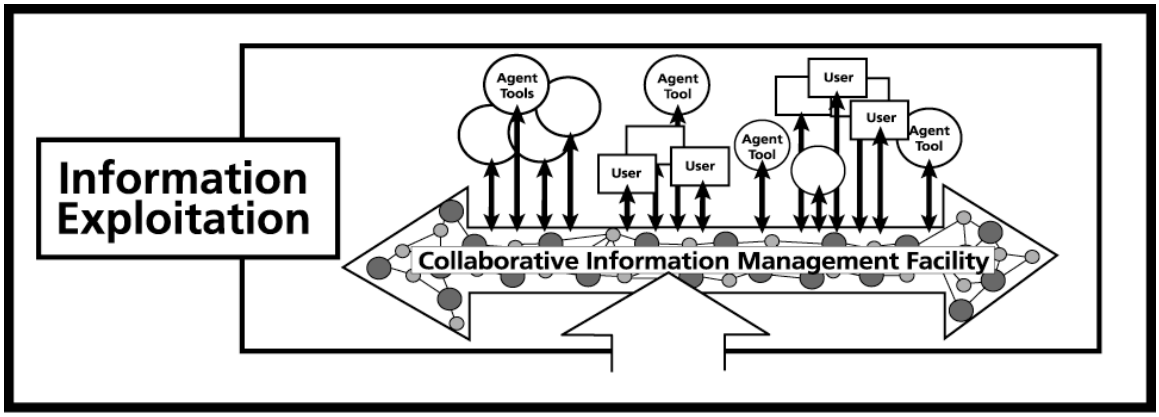


Figure 16d: The Information Layer of CICE

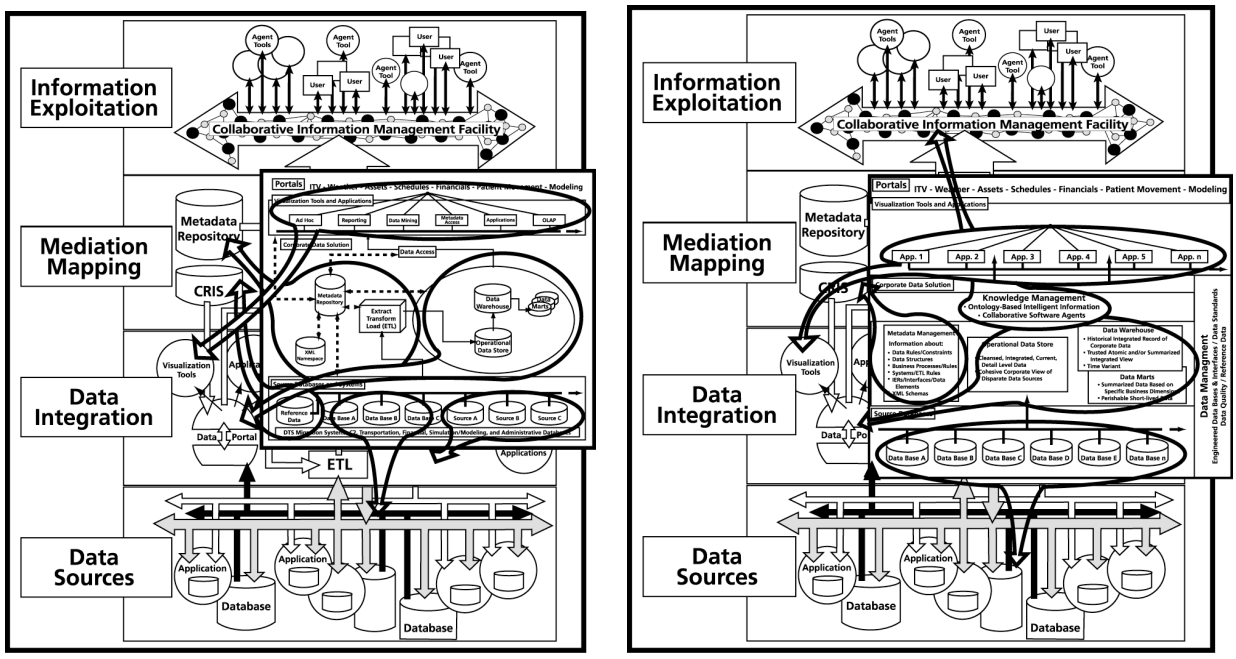
There are two main reasons why there can be many ontologies in any data domain defined by a single logical data model. First, particular data items are often utilized in many different ways. For example, a military transport asset such as a fixed wing aircraft may be involved as a weapon platform in a tactical mission, or as a conveyance in a logistical role, or as equipment requiring repair in a maintenance role. The same data entities may therefore appear within quite different contexts, represented by different relationships in three different ontologies within three ontology-based applications. For these ontology-based applications to be interoperable at the 'information' level it is essential that each of the ontologies comply with the underlying logical data model. In this way, the agents in one application are able to act on warnings and alerts received from agents in either of the two other ontology-based applications. For example, this will allow the Repair agent in the 'maintenance' application to automatically receive and interpret an alert from the Electrical Systems agent in the 'logistical' application that an intermittent malfunction has occurred during the last flight of a particular aircraft. The Repair agent is able to automatically consider this information in prioritizing the maintenance schedule of this aircraft.

Second, even within the same functional area there may exist multiple perspectives and therefore multiple ontologies (see Section 4.3 for an extended discussion of perspectives and facades).

Relationship to CDE architecture: The extension of the CDE architecture into the more comprehensive knowledge management environment proposed by the CICE architecture (Figure 16) involves some minor structural changes to the existing CDE architecture shown in Figure 14 (see Section 3.1). Specifically, the division of CICE into four architectural layers provides an opportunity for the principal components of the existing CDE architecture (with and without a knowledge management component) to be relocated according to their nature and purpose in the CICE architecture (Figure 17).

As shown in Figure 17(a) the Portals, Operational Data Store, Data Warehouse with Data Marts, and the data-centric applications are logically moved into the Data Integration Layer of CICE. All of these components are concerned either with the storage and visualization of data or the analysis of data. The Metadata Repository, ETL rules, and XML Namespace provide data schema definitions and business rules descriptions, and are therefore moved to the Mediation Layer. The same relocation is suggested for the Standard Reference Tables since they serve a

similar purpose to metadata and are maintained by database management staff. However, the various databases remain in the Data Source Layer.



View (a) without KM component View (b) with KM component

Figure 17: Merging the existing CDE architecture with the CICE architecture

In Figure 17(b) the same relocation of CDE components is shown in the context of the knowledge management (KM) component that was recently proposed for addition to the existing CDE architecture. This component is moved into the Information Layer of CICE.

As can be seen in Figure 17 the merging of the existing CDE architecture with the proposed CICE architecture involves only some reorganization, while preserving the principal components of the existing CDE.

4.3 Interoperability at the 'Information' Level

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

Past approaches to interoperability have basically fallen into three categories. Attempts to create common architectures have largely failed because this approach essentially requires existing systems to be re-implemented in the common (i.e., new) architecture. Attempts to create bridges

between applications within a confederation of linked systems have been faced with three major obstacles. First, the large number of bridges required (i.e., the square of the number of applications). Second, the fragility and inextensibility associated with hard-coded inter-system data linkages. Third, the cost of maintaining such linkages in a continuously evolving information systems environment. The third category of approaches has focused on achieving interoperability at the interface boundary. For anything other than limited presentation and visualization capabilities, this approach cannot efficiently accommodate dynamic data flows, let alone constant changes at the more useful information level. These obstacles to interoperability and integration are largely overcome in an information-centric software systems environment by representing adequate context to support automated reasoning capabilities. An integral part of such a capability is the inclusion of a subscription service that allows the interests to be described in the explicit terms of the information model(s). The technology that is most commonly employed to achieve this level of representation and 'understanding' in a software application is an ***ontology***.

An ontology in this sense can be defined as a constraint, abstraction and relationship rich model (or set of models) describing the entities, concepts, and notions relevant to the domain of operations. The problem arises when two or more of these systems, each operating over a potentially extensive set of descriptions attempt to collaborate with each other. While collaboration within each of these systems may be based on very high-level descriptions, it will undoubtedly be subject to various application-specific biases.

For example, in a tactical command and control system an entity such as a M1A1 tank may be viewed, and therefore represented as a tactical asset. In this case the bias would be toward tactical utility. However, in a logistics system the same M1A1 tank would most appropriately be viewed as a potential supply item with emphasis on logistical inventory and supply. In both cases, however, the subject is still the exact same M1A1 tank with its basic inherent characteristics. The difference resides in the manner in which the tank is being viewed by each of these systems. Another term for this bias-based filter is ***perspective***.

Perspective is not only a natural component of the way in which we perceive the world but moreover should be viewed as a highly beneficial and desirable characteristic. Perspective is the ingredient in an ontology-based decision-support system that allows for the accurate representation of domain-specific notions and bias. For example, if a decision-support system is to assist in the formulation of logistical supply missions then it is more appropriate, and beneficial for an entity such as a howitzer to be primarily viewed as a supply item instead of a tactical asset. If viewed as a supply item the description of the howitzer could provide significant detail in terms of the item's shipping weight, shipping dimensions, tie-down points, and so on, all of which is pertinent information for the transportation of the howitzer. In the context of a tactical command and control system, however, such information is of much less (if any) interest. Information that would be relevant in such a tactical system would be tactical characteristics such as projectile range, effective casualty radius, advancement velocity, etc. Conversely, these characteristics are of little or no significance in the domain of logistics. Nevertheless, regardless of the perspective it may be the exact same howitzer that is being discussed between the two disparate systems (i.e., logistics system and tactical system). However, it is being discussed within two different contexts exhibiting two distinctly different perspectives. While collaboration within or across systems supported by the exact same

perspective-based representation performs well, the problem arises when collaboration needs to occur between systems or system components where the perspectives are in fact not the same and potentially drastically dissimilar. In this unto common case, the extent to which systems can effectively collaborate on events and information, without a means of translation, is essentially limited to low-level data passing with receivers having little or no understanding of content and implication. Simply stated, the problem at the heart of interoperability between decision-support systems resides in the means by which information-centric systems exhibiting wholly, or even partially disparate perspectives, can interoperate at a meaningful and useful level.

The solution to this dilemma can essentially take two different directions. The first of these paths focuses on the development of a 'universal' ontology. Such an ontology would represent a single view of the world covering all relevant domains. Each system would utilize this representation as the core informational basis for operation. Since each system would have knowledge of this common representation of the entities, notions, and concepts, interoperability at the information level would be clear and concise requiring no potentially context-diminishing translation. However, as straightforward as this may appear there are two major flaws with this approach. First, in practicality it is highly unlikely that such a universal description could actually be successfully developed. Considering the amount of forethought and vision this task would require, such an undertaking would be of monumental scale as well as being plagued with misrepresentation. Inevitably, certain notions or concepts would be inappropriately represented in a particular domain in an effort to model them adequately in another.

The second flaw with the universal ontology approach is less obvious but perhaps even more limiting. Considering the number of domains across which such an all-encompassing ontology would need to extend, the resulting ontology would most likely be comprised mainly of generalities. While useful for some types of analyses these generalities would typically only partially represent the manner in which any one particular system wished to *see the world*. In other words, due to the number of perspectives a universal ontology would attempt to represent, the resulting ontology would ironically end up being just the opposite, a perspective-absent description essentially devoid of any domain-specific detail and falling far short of system needs and expectations. While perspective was the cause of the original interoperability problem it is still a highly valuable characteristic that should not only be preserved but should be wholeheartedly embraced and promoted. As mentioned earlier, perspective is a valuable and useful means of conveying domain-specific notions and bias, which are crucial to information-centric decision-support systems. To omit its presence is to significantly reduce the usefulness of an ontology and therefore the effectiveness of the utilizing decision-support system(s). This coupled with the highly unlikely potential for developing such a comprehensive, inter-domain description of the world renders the universal ontology approach both unrealistic and wholly ineffective.

The second, more promising solution to interoperability between decision-support systems introduces the notion of a ***perspective filter***. Based on the façade design pattern (Buschmann et al. 1996, Fowler 1997) perspective filters allow core entities, concepts and notions accessible to interoperating systems to be viewed in a more appropriate form relative to each collaborator's perspective. In brief, the façade pattern allows for a certain description to be viewed, and consequently interacted with in a more native manner. Similar to a pair of infrared *night vision* goggles, overlaying a filter may enhance or refine otherwise limited information. In the case of

ontology-based collaboration this filter essentially superimposes a more perspective-oriented, ontological layer over the initial representation. The filter may not only add or modify the terminology and constraints of the core descriptions but may also extend and enhance it through the incorporation of additional characteristics. These characteristics may take the form of additional attributes and relationships as well as refining existing constraints, or even adding entirely new constraints. For example, Figure 18 illustrates the use of a logistically oriented perspective filter over a core description of conveyances. Note first that while the core conveyance ontology appears to represent only a limited amount of bias the effectiveness of perspective filters certainly does not require such a general core description. If the core ontology were heavily biased toward a foreign set of perspectives it would simply mean that the perspective filters would need to be more extensive and incorporate additional constraints, extensions, etc. However, for clarity of illustration a limited, rather general core ontology has been selected.

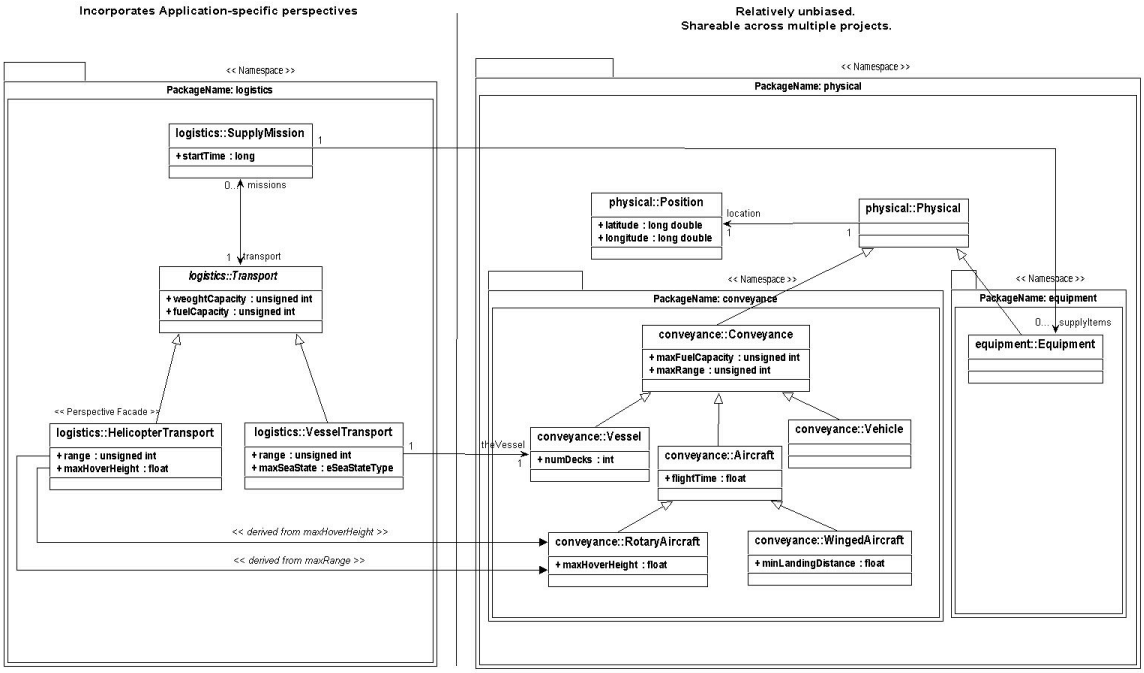


Figure 18: Partially derived logistics ontology

Core of the logistics perspective presented in Figure 18 is the notion of a transport. However, although the logistics system may have a notion of all of the types of conveyances (i.e., vessels, vehicles, and aircraft) represented in the core ontology, in the context of this example, it may only consider vessels and rotary aircraft as potential transports. In this situation it would be valuable to represent this refined constraint in the ontology forming the representational heart of the logistics system while still employing the core conveyance ontology. As Figure 18 illustrates, representing such refinement can be accomplished by explicitly introducing a constrained notion of a transport in the application-specific filter ontology. An abstract *Transport* is defined to have two specific derivations (*VesselTransport* and *HelicopterTransport*). At this point it is immediately apparent that a vehicle is not a transport candidate. In the context of the example

logistics system transports can only be *VesselTransports* or *HelicopterTransports*. The task now becomes linking these two system specific notions to the core conveyance ontology.

Relating these two transport types to their conveyance ontology counterparts can be achieved in two different ways. For illustration purposes, the definition of *VesselTransport* adopts the first method while *HelicopterTransport* employs the second. The first method defines an explicit relationship between the *VesselTransport* and the core description of a vessel outlined in the conveyance ontology. Utilizing this approach, obtaining the core information relative to the corresponding *Vessel* from a *VesselTransport* requires both knowledge of their relationship in addition to another level of indirection. For reasons of performance and representational accuracy, both of these requirements may be undesirable.

The second method, illustrated in Figure 18 using *HelicopterTransport*, avoids both shortcomings inherent in the first approach. In this case, *HelicopterTransport* exists as a façade, or filter, which transparently links at the attribute level into the core *RotaryAircraft* description. That is, each attribute of *RotaryAircraft* desired to be exposed to users of *HelicopterTransport* is explicitly declared in the façade. For example, since the maximum range of travel is relevant to the definition of a *HelicopterTransport* the *maxRange* attribute of *RotaryAircraft* (inherited from *Conveyance*) is subsequently exposed in the HelicopterTransport façade description. By virtue of being declared in a façade any access to such an attribute would be transparently mapped into the corresponding attribute(s) on which it is based. In the case of the *range* attribute of *HelicopterTransport*, access would transparently be directed to the inherited *maxRange* attribute of *RotaryAircraft*. Notice also the use of alternative terminology over that used in the core ontology (i.e., *range* vs. *maxRange*). It should also be noted that the derivative nature of a façade attribute is not limited to mapping into another attribute. Rather, the value of a façade attribute may also be derived through calculation, perhaps based on the values of multiple attributes residing in potentially several different core objects. In either case, the fact that the value of the façade attribute is derived is completely transparent to the façade user.

Another perspective-oriented enhancement to the core ontology illustrated in Figure 18 is the notion of a *SupplyMission*. Being a fundamental concept in the example logistics system a supply mission essentially relates supply items in the form of equipment to the transports by which they will be delivered. Once again, the definition of a logistics-specific notion (i.e., supply items) is derived from a notion defined in the core ontology (i.e., equipment). In this case, an explicit relationship is declared linking *SupplyMission* to zero or more *Equipment* items. Since, from the perspective of the logistics system *Equipment* scheduled for delivery are viewed as items that are to be supplied, the term *supplyItems* is used as the referencing nomenclature. Such an enhancement demonstrates the ability to integrate new concepts (i.e., supply missions) with existing core notions.

Considerable advantages accrue from drawing relevant concepts and notions into a system's local set of perspective-rich, filter ontologies. As the above example illustrates, key components of these perspective-oriented ontologies could be derived from a set of core, relatively unbiased common notions forming the basis for informational collaboration among systems. There are several benefits to adopting this approach. Collaboration among information-centric, decision-support systems would take place in terms of various core ontologies (i.e., *Conveyance*) with each collaborating application viewing these core entities, concepts and notions according to its own perspective. Figure 19 briefly extends the logistics example presented in Figure 18 showing

collaboration between the original logistics system and a tactical command and control system. Collaboration between these two example systems is in terms of the common, core ontologies on which they share their derivations. A conveyance is still a conveyance whether it is viewed in the context of logistics or tactical command and control. To represent domain-specific notions (e.g., transport, supply item, tactical asset, etc.) each collaborating system would apply the appropriate filter. Although discussing a conveyance from partially disparate perspectives both systems can collaborate about core entities, concepts, and notions.

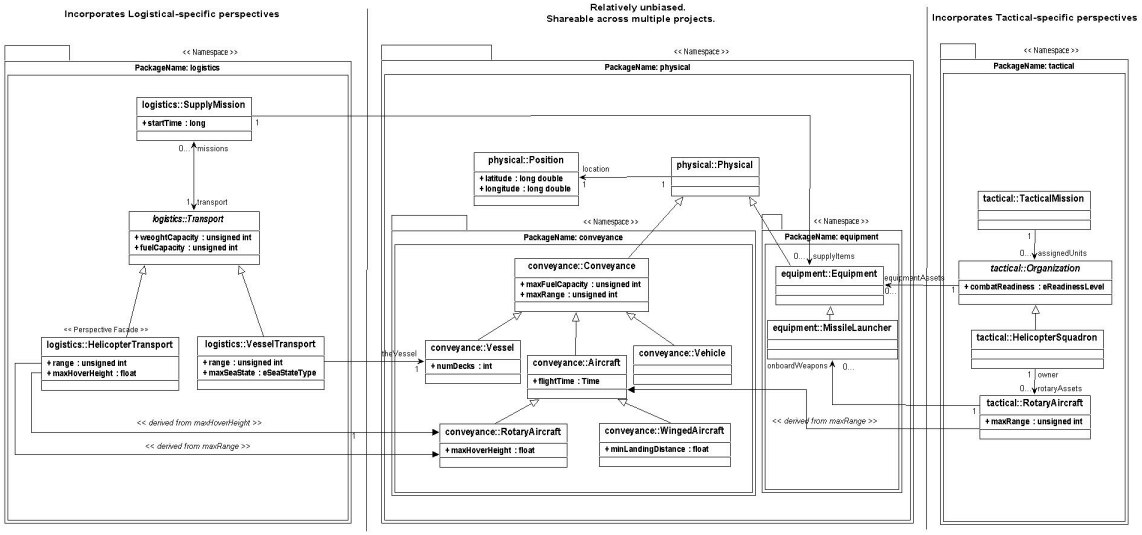


Figure 19: Two disparate domains linked into the same core ontology

Another advantage of supplementing core, non-system-specific ontologies with perspective rich filters is the preservation of both time and effort during the development of such information-centric systems. Core ontologies could be archived in an ontology library forming a useful reference source for the development of new system ontologies. Models created for new decision-support systems could make use of this ontology library as a strong basis for deriving system-specific filters. In addition, such a process would promote the use of common core descriptions increasing the potential for interoperability even further.

4.4 Design Principles for Ontology-Based Applications

As USTRANSCOM transitions to a knowledge management environment (i.e., CICE) there is an opportunity, and arguably a compelling need, to establish software design principles and guidelines that will support the kind of interoperability at the 'information level' described in the previous section (see Section 4.3). This is particularly important during a major software transition effort of this kind. Key design principles that should be incorporated in any new ontology-based software applications that will form part of CICE's Information Management Layer are collaboration-intensive, context-based representation, flexibility and adaptability, and multi-tiered, multi-layered architecture (Figure 20).

Collaboration-Intensive: Certainly in the real world, collaboration among decision-makers and experts is a critical ingredient in making educated and effective decisions. This is especially true when operating across an extensive and varied set of domains. The same quality extends to the realm of agent-based decision-support systems. Conceptually, such systems consist of dynamic collections of collaborators (both human and software-based) each playing a role in the collective analysis of a problem or situation and the consequential decision-making assistance required in formulating an accurate assessment and/or solution.

Whether human or software-based, collaboration within an ontology-based system occurs in terms of a descriptive ontology (Chandrasekaran et. al. 1999). Until recently these ontologies were limited to describing information and knowledge that represents various aspects of the domain(s) over which the system is to operate.

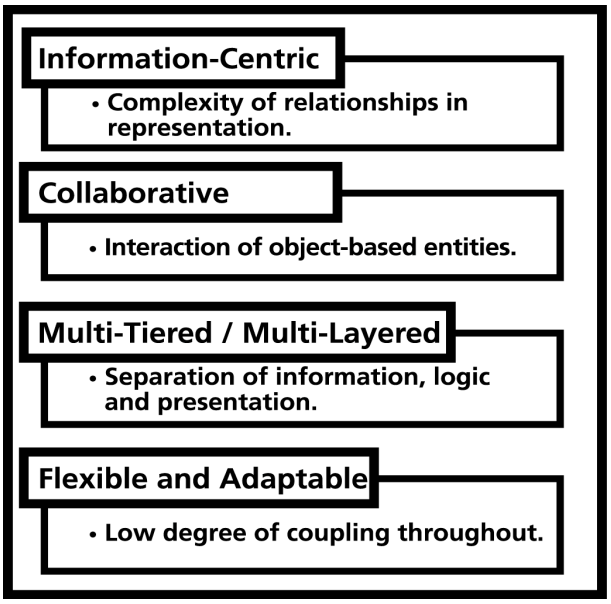


Figure 20: Design principles for Ontology-based applications

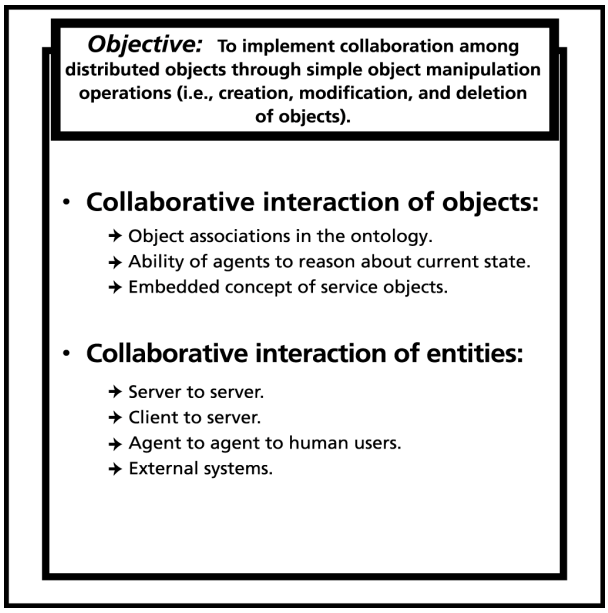


Figure 21: Design principle: *Collaboration*

For example, in the domain of architectural design the applicable ontology would describe such notions as spaces, walls, accessibility, appropriate lighting, and so on. Although effective and certainly a fundamental element of an information-centric system a considerable portion of the system still remains in a form not necessarily supportive of highly collaborative environments. Further, these non-ontology based components require separate and dedicated interfaces along with specialized management. A number of the services that collaborators within an ontology-based system interact with (e.g., time, query, subscription, execution, reasoning, etc.) were still presented as client-side adjunct-based interfaces requiring additional management to support collaboration. For example, if two clients wish to share or discuss the same subscription profile, a separate mechanism for identifying and referencing the collection of interests is required. In this case, the interface would be the client-side Application Programming Interface (API) maintained by the subscription service itself. Although certainly possible, supporting such specialized functionality requires the particular services (i.e., the subscription service in this

case) to present and manage a specific API to expose or match global references. Although subtle in nature, complexity such as this can easily escalate when considering the high degree of collaboration inherent in multi-agent decision-support systems.

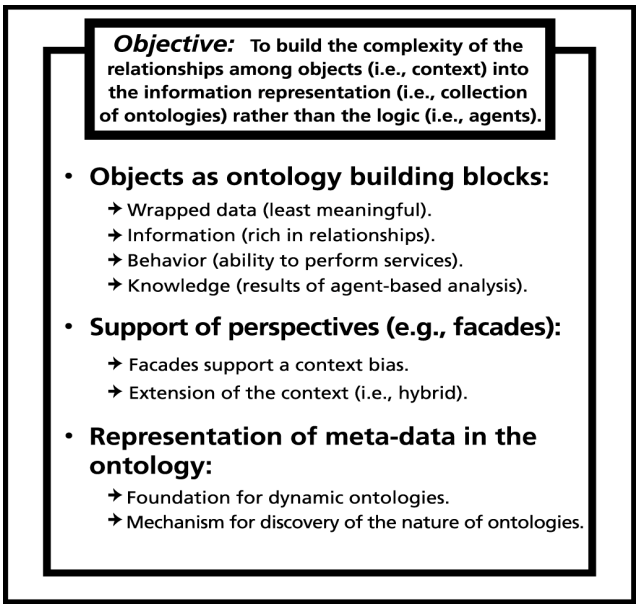


Figure 22: Design Principle: *Information-`Centric (Context)*

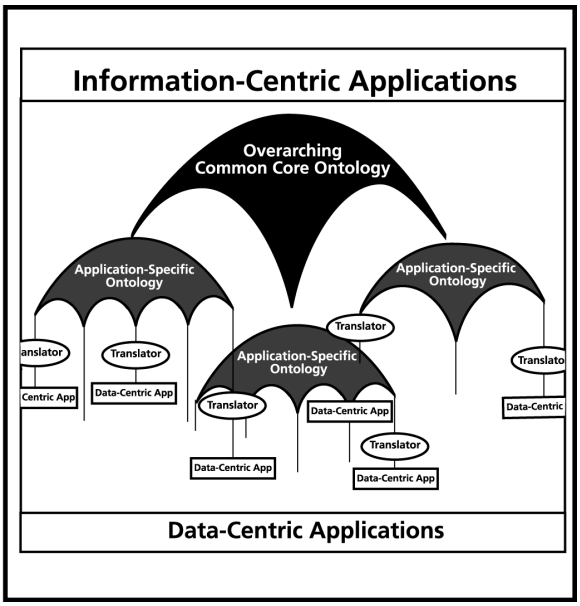


Figure 23: Ontology cluster supporting multiple levels of abstraction

This limitation can be overcome by taking the notion of *objectified collaboration* to the next level (Figure 21). This approach extends the once solely information-based ontology to also include the behavioral aspects of the decision-support system. More specifically these constrained behavioral objects constitute the services within the decision-support system (i.e., the services themselves are represented in the collaborative ontology in the same manner as information and knowledge). The only difference is that these distributed and shareable objects offer behavior in addition to information. As a result collaborators are able to interact with these services through the same distributed object operations that they would perform on the information and knowledge objects. Any constraints identified in the behavior are enforced by the standard ontology management facility. The operations that can be performed on these ontology-based objects consist of the basic creation, deletion, and modification functionality. To support the aforementioned example in which two collaborators wish to reference and discuss aspects of the same subscription profile, the two collaborators would treat the profile in question as just another set of multi-faceted, shareable distributed object. In other words, similar to the manner in which rich information models or ontologies are used as a basis for collaboration, this notion is extended to include interaction and collaboration across the services that constitute the system itself. The effect is essentially that interaction with and collaboration across information and now behavior (e.g., services), is reduced to a basic set of object manipulation capabilities. In this sense, an object is-an-object is-an-object. The only difference is that some distributed, shareable objects offer information while others may also offer behavior. The client-side portion of the ontology replaces the need for specialized client-side functionality.

Context (Information-Centric): Representation can exist at varying levels of abstraction (Figure 22). The lowest level of representation considered in this paper is *wrapped* data. Wrapped data consists of low-level data, for example a textual e-mail message that is placed inside some sort of an e-mail message object. While it could be argued that the e-mail message is thereby objectified it is clear that the only objectification resides in the shell that contains the data and not the e-mail content. The message is still in a data form offering a limited opportunity for interpretation by software components.

A higher level of representation endeavors to describe aspects of a domain as collections of inter-related, constrained objects. This level of representation is commonly referred to as an information-centric ontology. At this level of representation context can begin to be captured and represented in a manner supportive of software-based reasoning. This level of representation (i.e., context) is one of the most empowering design principles that can be applied in an ontology-based application. Further, as mentioned in the previous section portions of this context may be extended to exhibit behavior. In addition to services, however, distributed behavioral objects can also be employed as a mechanism for supporting the notion of facades (useful in supporting alternative perspectives in addition to other applications (see Section 4.3)).

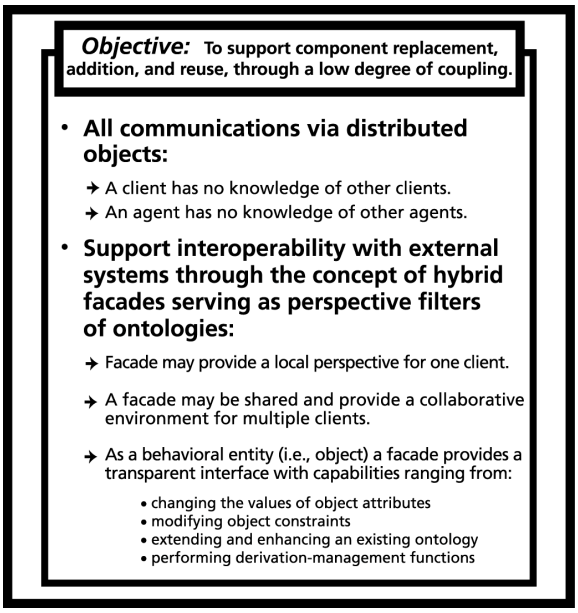


Figure 24: Design principle:
Extensibility and Adaptability

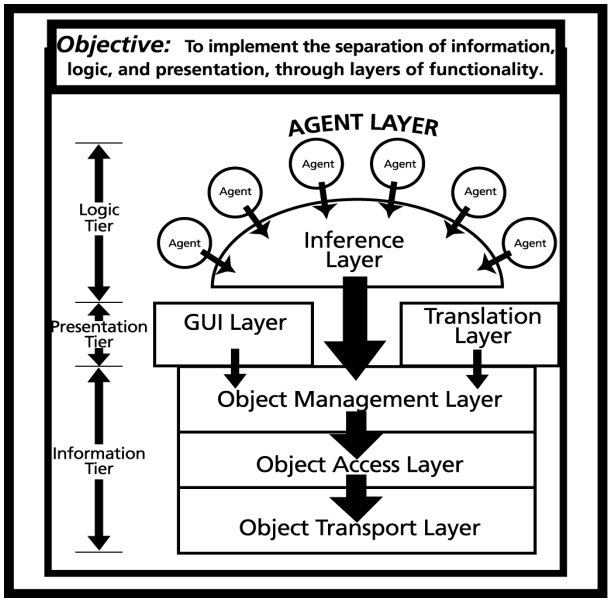


Figure 25: Design principle:
Multi-Tiered, Multi-Layered Architecture

Existing as one of the fundamental design patterns employed in object-oriented design facades provide a level of derivation attained from the particular representation or ontology on which they are based (see Section 4.3). Facades offer a method of supporting and managing an alternative *perspective* to the native description from which they are derived (Pohl 2001). In other words, facades allow the perspective inherent in a particular model of a domain to be augmented, or in some way altered to support a more appropriate (i.e., to the façade user) representation of the concepts, notions, and entities over which that *user* is operating (Figure 23).

Note that *user* in this sense refers to any accessing component. While certainly useful in systems supporting multiple perspectives caution must be employed in preventing abuse by introducing inconsistency and unnecessary duplication.

Facades can also be utilized to support real-time calculations. In this sense, the façade derivation would involve a calculation or algorithm perhaps based on one or more attributes of the base object(s). For example, consider an architectural space exhibiting length, width, and height described in English standard units, which is to be accessed by a design system that only understands metric units and also requires space volumes. Utilizing ontology-based facades a model could be developed in which, not only the length, width, and height, but also the volume of the space could be calculated and presented to the design system in terms of metric units. Although there are a number of approaches to supporting calculated attributes in the case where an alternative perspective is to be supported, the façade approach permits an extensible (i.e., one perspective extended from another) and encapsulated (i.e., easily maintainable) solution.

Extensibility and Adaptability: A primary goal of an ontology-based system should be a high degree of flexibility in respect to the configuration of its components both at the development and execution levels. Such a system should support the addition, replacement, and reuse of software components and achieve this goal by reducing inter-component coupling to an absolute minimum (Figure 24). There are two key system properties that permit this kind of flexibility. First, all collaboration between clients should take place via, and in terms of the informational ontology (i.e., distributed objects). No direct communication should exist between collaborators (this is not to say that specific inter-client communication could not be explicitly modeled in the ontology). The result is a collaborative environment in which client identities are essentially irrelevant in respect to this process. This low degree of coupling permits the reconfiguration (i.e., component addition, removal, or replacement) of collaborating components at any point during an execution session.

The second property deals with the manner in which clients access and interact with the ontology. For example, an ontology-based software development environment might offer a standard interface component such as an object management layer, which both shields accessors from the complexity of ontology management as well as provides an abstracted view of the ontology. Clients of this object management layer interact with the ontology via object wrappers based on a set of corresponding ontology-specific templates. Promoting the notion of adaptability, these templates are discovered by the object management layer as a runtime activity. The resulting support for dynamic definition permits elements of the ontology to be extended, eliminated, or even redefined during the course of a runtime session. Although such a dynamic definition capability provides considerable flexibility, it does bring into the equation the potential for conflicting definitions and breaching of the informational integrity of the system. As such, this capability needs to be well thought out and incorporated in a semantically sound manner.

Apart from the ability to adapt to an evolving definition of a domain, adaptability should likewise be supported in interaction with external systems. This level of adaptability functions in conjunction with the concept of façades mentioned earlier. Replacing the classical approach of building a dedicated and separate translation bridge between collaborating systems, the translation may be incorporated into the ontology itself. In other words, support for ontology-

based facades allows the translation or derivation of each system's perspective to be encapsulated and managed primarily within façade objects. The resulting translation facility exists as a set of behavioral façade objects accessed and manipulated in a manner no different than is applied to other ontology objects. The result is an elegant design where support for translation-based communication between disparate systems is seamlessly incorporated as part of the ontology.

Multi-Tiered and Multi-Layered: The forth design principle addresses the architectural organization of ontology-based systems. More specifically, this principle identifies distinct separations between areas of functionality at both the conceptual (i.e., tier) level and the more concrete (i.e., layer) level. Conceptually, the architecture of an ontology-based decision-support system is divided into three distinct tiers namely, information, logic, and presentation. To manage its particular domain each tier contains a number of logical layers that work essentially in sequence (Figure 25).

As the name suggests the information tier houses both the information and knowledge (i.e., ontology) being operated on in addition to all of the mechanisms needed to support management, transport, and access. The information is further delineated into layers. The first of these is the object management layer described previously. Below the object management layer resides an object access layer responsible for managing access to the information tier. The object access layer exists as a level of abstraction below the object management layer and interfaces directly with the object transport layer. In the case of a CORBA (Common Object Request Broker Architecture) environment (Mowbray and Zahavi 1995) the object transport layer would be responsible for communicating the various requests and subsequent replies for distributed information and behavior issued through the object access layer throughout the system. The object transport layer is the only layer that forms a dependency on an underlying communication protocol. As such, support for alternative communication facilities can be implemented with minimal impact on either the object access layer or the object management layer. This is an example of the benefits of a layered architecture in supporting component reuse and replacement.

The Logic Tier contains the business rules (i.e., agents) and analysis facilities by which these rules are managed. Different kinds of reasoning methodologies, such as opportunistic rule-based analysis, may be applied. Regardless of which form of reasoning is employed this capability is supported by two layers namely, the business rule layer and the business engine layer. The business rule layer is primarily system-specific and contains the agent-based analysis facilities resident in the system. Execution of agents is in turn managed by the business engine layer. To integrate the Logic Tier with the Information Tier the business engine layer interfaces with the object management layer permitting the agents to both access and contribute to the ontology.

The final tier is the Presentation Tier. This tier is responsible for interfacing with the various users of the system. In this sense a user may be a human operator or an external system. In the case of a human operator support is typically provided through a graphical user interface layer that presents and promotes interaction with the contents of the Information Tier. In the case of an external system, support takes the form of a translation layer that manages the mapping of representations between systems. Like the graphical user interface layer, access to and from the Information Tier is supported by the object management layer.

4.5 Use-Case Centered Development Process

It could be argued that it is *not* the goal of either the existing CDE or the proposed CICE to bring together all of the data in USTRANSCOM, nor to create a single data model for every system and application. These are essentially implementation details. To concentrate on these aspects is to lose sight of the fact that the principal purpose of any system is to provide value to end-users. Several questions then arise: Who are the end users? What decisions do they need to make? How would access to integrated data and information (i.e., data in context) help in the decision process? In general, these are all aspects of a single question: If users had access to all of the data in all of the systems used anywhere in USTRANSCOM, what would they want to do with it?

These questions are complicated by the fact that CICE is bound to change the way in which users perform their work, and will certainly create the possibility that they will be doing different kinds of work than they do now, once the system is in operation. As a result, it is impossible to determine what the real requirements of the system will be once it is built, because no one can precisely predict what kinds of changes are likely to take place.

The solution lies in an ***iterative development*** process. The guiding principle of iterative development is to deliver functional software at short intervals to end-users who then provide feedback and guidance on future requirements. The process of defining requirements becomes incremental, and the basis of collaboration between end-users and system designers. Since end-users know how they perform their work under current conditions, they must be considered an important source of input for defining implementation priorities. While designers and developers can foresee future possibilities, they typically cannot predict whether a given piece of functionality will in fact become an important capability for the end-users. Both have knowledge that can guide development, and both are necessary for a successful system. In order for this concept to be realized, it is important to stay focused on the needs of the users. This approach is often referred to as ***use-case centered*** development.

There are many forms of use-cases, differing primarily in their relative formality (Cockburn 2001). Basically, a use-case is a story that tells how an ‘actor’ will interact with the ‘system’. Actors can be either human users or other systems. The ‘system’ can be either the entire system or just a part of a system, depending on the objectives and role of the ‘actor’ for a given use-case. Use-cases can provide the basis for requirements discovery and definition. As such, they describe the actor's view of the system under discussion. Use-cases describe the behavior of the system given input from the actor, but only that behavior that the actor is aware of (plus important side effects, if any). However, use-cases do not include details of system implementation or internal design such as data models. They also do not describe the user interface.

A complete use-case includes alternate paths (referred to as ‘extensions’), which describe all the situations under which either the actor or the system can perform different actions based on the current state. The use-case also includes failure scenarios (i.e., conditions under which the system is not able to support the user's goal), along with pre-conditions (i.e., what must be true before the use-case can be executed) and guarantees (i.e., what will be true after the use-case has been successfully executed). Each use-case constitutes a contract for the behavior of the system. To facilitate the implementation of CICE, it would be very helpful to draw up use-cases that describe how the CDE users currently accomplish their goals. Knowledge of current planning and decision-making processes will provide invaluable information to software developers and

program managers to determine the data sources and information models (i.e., ontologies) that will be needed in order to implement these existing use-cases in the new knowledge management environment.

Use-cases and iterative development: A set of use-cases can form the starting point for a development process. In an iterative development process, the system is implemented incrementally and delivered to end-users as soon and as often as possible. As users receive successive versions of the software, their responses frequently result in new or modified use-cases, which must be incorporated in future iterations. New requirements are discovered as users and developers work with the system.

This is in contrast to processes that attempt to specify all the requirements before development begins. Comparatively, iterative development processes tend to produce systems that are more accepted by users, since developers are able to respond to changing goals and needs as implementation progresses. This characteristic is especially important in a system like CICE, which is likely and intended to change the way that people perform their work. Requirements for such a system will evolve as users see new possibilities.

For the implementation of the Information Layer of CICE, use-case oriented iterative development should begin by identifying major stakeholders and user categories. For each use category, it is important to find a representative user to provide input and perspective. The initial impetus for building the existing CDE was undoubtedly at least partly driven by a desire to reduce, and if possible eliminate, the need for planning staff and decision makers to manually bring together data from multiple existing systems in order to accomplish their goals. If this is the case, it is vital to include representatives of these planners and decision makers in the initial group of stakeholders and users.

The process of discovering use-cases for CICE will begin by listing examples of situations requiring multiple data sources. Each example should include the reason for bringing together these data, the list of data sources, the method of data extraction (e.g., existing client applications, direct database queries, etc.) and the type of data retrieved from each source. From this information, an ***as is*** use-case can be defined, followed by the corresponding ***to be*** use-case, which describes the user's interaction with the planned system.

The initial set of use-cases should be prioritized to ensure that the most important interactions are implemented early in the project's lifetime. The criteria for use-case prioritization are primarily user-oriented (e.g., How often does the user need to execute this use-case? How much time does it take to gather the data? How significant is the result likely to be?). However, especially during the early stages of the development cycle, developer priorities are also critical. Use-cases may require building significant parts of the planned system architecture, or they may involve parts of the architecture that developers see as risky. Both of these situations would cause a use-case to assume a higher priority from a developer's point of view, since the architecture should be built as soon as possible and potential risks should be addressed early in the project when alternatives are still available should the risk prove insurmountable.

Priorities may also be affected by the data sources involved in a use-case. To the extent possible, each use-case implemented should include one or more data sources that have already been integrated into the system in earlier use-cases, and in addition should include a data source that is

not yet part of CICE. In this way, successive development iterations will build on previous work, while gradually extending the range of integration.

Once the first set of use-cases has been prioritized, developers will determine how many use-cases can be reasonably implemented in the first development cycle. Due to the complexity of integrating new data sources into CICE, it is likely that the first cycle will be somewhat lengthy; – possibly as much as six months long. This duration must be estimated in large part by the data source integration team, based on the specific data sources involved. The first cycle is likely to consist almost entirely of building integration and architectural infrastructure, but will also include a (possibly small) number of use-cases. The key goal of all iterative development processes is that at the end of each development cycle, there will be releasable software. Whether a particular version of the system is released for use or not is a decision that is likely to be made outside the development team, but each development cycle should result in a system that can be released if that decision is made. In the case of CICE, the product of the first cycle should be released at least to the user groups whose use-cases are included in the system.

Later development cycles will follow essentially the same pattern. As users work with the evolving system, they will generate new use-cases and extensions to existing use-cases. Some older use-cases may become obsolete as the new system changes the way that users are working. For the Information Layer of CICE, it is likely that developers will see ways that some use-cases can be modified by constructing agents to determine that a user may benefit from specific information. And, as more data sources are added to the Operational Data Store or Data Warehouse, new types of processing (e.g., OLAP and Data Mining applications) may become both possible and useful. This, in turn, will also increase the number of potential use-cases.

At the beginning of each development cycle, new and old use-cases will be prioritized together, and developers will again determine which ones can be attempted for the next release. Developer priorities for subsequent cycles will include looking for use-cases that allow them to extend functionality created for prior cycles. After the first cycle, the time between releases can be shortened so that users can see the system evolving quickly. This increases the chances of user acceptance, since the effects of their requests for change can be seen over a relatively short period of time.

As the functionality of the system progressively increases, new user groups can be included. These might include, among others, the users of some of the systems that will feed information into CICE. These users might benefit from access to a wider range of data and information than is provided by the system they are currently using. There may also be opportunities to simplify or enhance their work, through the use of information layer capabilities on top of the same data that they currently use.

Iterative use-case centered development processes tend to produce software systems that are accepted by end-users, for several reasons. First, the end-users themselves are directly involved in defining requirements. Second, end-users see the system at an early stage and as it evolves. At each release, users have an opportunity to correct the direction that the development team is moving, and to add new requirements. Third, the requirements implemented during each development cycle are the highest priority, based on the input of all stakeholders, including the users themselves. Together, these aspects of iterative development ensure that at any point in time, the system meets the most important user needs.

5. CICE Implementation Recommendations

The business case for extending the CDE to a fully capable knowledge management environment, as proposed by CICE, has been discussed previously in Section 4.1, and will not be reiterated here in any detail. However, in summary the case is centered on the more efficient and appropriate use of diminishing human resources, the need for more timely responsiveness under expanding responsibilities, and the increasing economic burden of a data-centric IT environment with many potential failure points.

The principal impediment of USTRANSCOM's existing data-centric enterprise architecture and CDE is the reliance on human resources to interpret and manually relate to the appropriate context the vast amounts of data that come from a proliferation of heterogeneous data sources. This is the core problem that essentially generates all of the undesirable symptoms that severely restrict USTRANSCOM's ability to efficiently execute its transportation planning and execution responsibilities. As summarized in Table 1, a strong business case can be made in support of the need for USTRANSCOM to expeditiously transition from a data-centric to an information-centric enterprise IT architecture.

Table 1: Summarized business case in support of CICE

Alignment with DoD vision: The GIG vision of a global network of seamless connectivity is based on the concepts of data collection in context, automated data interpretation, data discovery, and intelligent decision-support. The realization of these concepts requires an information-centric environment capable of supporting intelligent tools (i.e., software agents) with reasoning capabilities.
Diminishing human resources: The manual interpretation and manipulation of an increasing volume of data is a wasteful use of valuable and scarce human resources. Realization of the GIG vision will increase the volume of collectable data by several orders of magnitude. Without the automated data filtering and interpretation capabilities of an information-centric architecture the GIG would be destined to choke under an enormous volume of raw data.
Changing military context: The changing military context is placing an unprecedented emphasis on responsiveness, interoperability, adaptability, and accuracy. A data-centric environment that requires the human operators to be engaged at the least productive level of data manipulation, cannot respond effectively to the information and decision-support needs of the commanders.
Increasing maintenance costs: An overloaded data-centric IT environment forces an organization to dedicate a significant portion of its operating budget, staff, project budgets, and time, on the piecemeal resolution of ad hoc problems such as data bottlenecks, transmission delays, breakdown of data exchange interfaces, and the inability to quickly find critical data.
Increasing system complexity: The increasing complexity of networked systems calls for automated, intelligent system management capabilities. Autonomic computing requires the context provided by an information-centric environment to automatically discover, diagnose and react to disruptions, maximize resource utilization, and anticipate, detect, identify, and mitigate external attacks.

5.1 Risk Mitigation

The efficient and expeditious implementation of CICE hinges on a web of interwoven technical and human factors. As a large and complex software development project, the success of CICE is predicated on the effective coordination of both the developers who will produce the tools that

are expected to greatly increase the capabilities of their users, and the users who may be reluctant to adapt their existing procedures to take advantage of those tools.

On the technical side, the risks associated with an IT project of this magnitude are largely related to process. In recent years the traditional software development process that proceeds in a strictly defined sequence of stages has given way to an iterative style of development that relies on a spiral model of planning, engineering and evaluation (Boehm 1986, Gilb 1988, Martin 1991, Stapleton 1997). The principal characteristics of a spiral software development process include: sound overall coordination and technical leadership; relatively small iterative steps; emphasis on quality at each step; frequent, usable products; accurate and meaningful progress monitoring and documentation; and, close interaction with the users (Palmer and Felsing 2002). ***To mitigate the potential technical risks it is strongly recommended that the principles of a spiral development process be applied to the implementation of CICE.***

On the human side, attempts to accelerate the normal evolutionary transition of a large organization from one state to another often fail due to the natural resistance to change of the majority of the members of the organization. This resistance is exacerbated by a very strong human survival instinct. 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 cling onto 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 any venture into new and unknown territory as a devaluation of our existing (i.e., past) experience.

The transition of the USTRANSCOM community from a data-centric information systems environment in which even the lowest levels of data interpretation, filtering and context-based analysis have to be undertaken by human users, to an information-centric knowledge management environment with intelligent collaborative tools, is a formidable undertaking. The level of success attained in this undertaking will depend largely on how well the organization is able to overcome human rather than technical obstacles. The technical capabilities to achieve the desired knowledge management goals have been available for several years and are maturing rapidly. Overcoming the human resistance to change, however, will require a major organizational effort involving an unprecedented degree of coordination, communication, and education. ***To mitigate the human risks, the most intensive involvement of the users of CICE in the design, implementation, and testing of the incremental products of the development process is highly recommended.***

5.2 Short Term Strategy (12 Months)

The implementation of the proposed knowledge management environment is predicated on the acceptance in principle of the CICE concepts and their integration into USTRANSCOM's enterprise architecture. Therefore, as a first step it will be necessary for the CICE architecture to be formally endorsed and incorporated in the Defense Transportation System Enterprise Architecture. This process should be expedited to the extent possible, with the expectation that it will be completed within less than six months. However, since this is largely a documentation

requirement, it should not delay immediate action on the short-term strategy recommendations that follow Recommendation (1) below.

Recommendation (1): Formally incorporate the CICE notions, principles, and architecture, in USTRANSCOM's Defense Transportation System Enterprise Architecture.

It is recommended that the short term strategy for implementing CICE focus on four parallel and related initiatives: (1) examination, extension and consolidation of the Mediation Layer; (2) exploratory software development aimed at both testing the new capabilities of the Mediation Layer and incrementally exposing user groups to the powerful new capabilities that CICE will provide; (3) formation of an organizational technical management instrument that will guide the design and implementation of CICE; and, (4) establishment of a multi-level user group structure that will ensure the intensive participation of the USTRANSCOM user community in the identification, prioritization, design, implementation, and testing of the new intelligent capabilities. These initiatives are closely related in as much as the expeditious and successful implementation of CICE will depend to a large extent on the willingness of its potential users to enthusiastically embrace its creation. Therefore, from the outset at least equal emphasis should be placed on user education, motivation and participation, as on the technical implementation aspects and issues.

1. The two principal innovations that will distinguish CICE from the existing CDE are related to the Mediation Layer and the Information Layer. The purpose of the Mediation Layer is to provide the necessary descriptions of the structure and nature of data to support the automated interpretation and mapping of data to the context-based information models that exist in the Information Layer. The innovations are therefore directed to the processing of data in context and the provision of more powerful tools that are capable of reasoning, with some level of intelligence, about the state of the resulting information-centric environment. In this respect the Mediation Layer is a prerequisite for the most important new capabilities of the proposed knowledge management environment.

Recommendation (2): Carefully examine the existing CDE components of the Mediation Layer to determine whether the current implementation of these components is capable of supporting a greatly extended mediation capability and what (if any) changes are required. In particular, assess the near real-time, on-line accessibility by applications and systems of the existing metadata registries and business rule repositories.

Recommendation (3): Review the existing process (i.e., principles, policies, practices, and periodic assessments) that has been implemented to coordinate and guide the evolution of USTRANSCOM as a Community of Interest (COI) within the DoD net-centric information systems vision (Stenbit 2003) and identify necessary revisions and extensions to this process.

Recommendation (4): Identify, design, and implement a set of tools in support of the process that will guide and facilitate the extension of the existing CDE to CICE. Candidates for such process facilities include: search and merge tools; Master Model compliance verification tools; ontology generation tools; real-time constraint checking tools; data-to-information mapping tools; data transformation tools; run-time management tools; testing and evaluation tools; and, autonomic computing tools.

Recommendation (5): Survey the current status of the metadata collection and registration effort, identify deficiencies, review the current incentives program, recommend measures for strengthening the incentives program, prioritize registration targets, and develop a plan for accelerating the collection and registration of metadata in the Mediation Layer of CICE.

2. There is a need to explore the ability of the Mediation Layer to support its critical role in CICE, using a 'test case' approach. In particular, it is necessary to verify the capabilities of existing tools and identify the need for additional tools, and quantify the constraints that may govern the performance of the Mediation Layer execution environment.

Recommendation (6): Utilize the existing migration system, ICODES (Integrated Computerized Deployment System), as a 'test case' for the development of a generic Master Model compliance verification tool. (ICODES is an information-centric ontology-based application that is owned by USTRANSCOM (via MTMC) as a migration system, and has been deployed to the Army since 1999 and the Marine Corps since 2002.)

Recommendation (7): Develop a new prototype ontology-based application as a 'test case' to explore the software engineering ramifications of adhering to all aspects of the process established for guiding the transition to CICE (e.g., near real-time access to metadata registries, Standard Reference Tables, business rules, XML Registry, and Ontology Registry).

3. An organizational technical management committee structure is required to guide the design and implementation of CICE. Responsibilities of this technical oversight instrument should include: identification and resolution of architectural issues; recommendations for process changes and additions; identification and resolution of metadata repository issues; determination of technical priorities; establishment of milestones; development of coarse granularity schedules; establishment and application of technical (i.e., system level) performance metrics; and, monitoring of technical progress.

Recommendation (8): Appoint a CICE Technical Committee (and sub-committees if appropriate) with representation on a CICE Steering Committee. Membership should include representatives from the various USTRANSCOM divisions and departments with architecture, data and related technical responsibilities. The possibility of including on this Technical Committee some technical expertise from an external source should be considered.

4. An organizational multi-level user group committee structure is required to provide an effective vehicle for the enthusiastic participation of the potential user community in the design and implementation of the functional CICE capabilities. Responsibilities of this functional oversight instrument should include: assignment of functional priorities; planning of an iterative development process for new functional capabilities; oversight of functional software tests and evaluations; resolution of functional issues related to CICE; recommendations for process changes and additions; identification of user support requirements; and, monitoring of functional progress.

Recommendation (9): Appoint a CICE Functional Committee (and sub-committees if appropriate) with representation on a CICE Steering Committee. Membership should include

representatives from the various USTRANSCOM user groups (i.e., divisions and departments with functional and operational responsibilities).

5. There is a need for a higher-level steering committee to oversee the CICE implementation program and ensure the integration of the functional and operational requirements and desires of the user community with the technical implementation capabilities and constraints. Perhaps most appropriately chaired by the USTRANSCOM CIO or his representative, the steering committee membership should include representation from the functional and operational (i.e., user community), as well as the technical data and system management branches of USTRANSCOM.

Recommendation (10): Appoint a CICE Steering Committee with representation from the CICE Technical Committee, the CICE Functional Committee, and other external organizations as deemed appropriate and necessary (e.g., USTRANSCOM Component Commands, DISA, etc.).

5.3 Longer Term Strategy

Within the very short period of time available for knowledge acquisition in preparation for this report (and the terms of the contractual statement of work) it was not possible to comprehensively review all aspects of USTRANSCOM's data and information systems environment. Accordingly, priority was assigned to a cursory review of documentation describing the existing DTS Enterprise Architecture, the USTRANSCOM Data Management Handbook, the GIG Architecture, and the DoD Net-Centric Data Strategy. Therefore, at this time, the authors of the report have insufficient knowledge of the major programs (e.g., AT-21, GTN-21, etc.) that are either in progress or planned, to suggest definitive steps for utilizing one or more of these programs as a vehicle for the implementation of CICE.

However, from a general perspective, it would appear appropriate to utilize one or more of these existing major contract vehicles for the expeditious implementation of CICE. In any case, to do otherwise would be counterproductive for several reasons. First, it would place the implementation of CICE into direct competition with parallel system implementation efforts that may not be compatible with the goals, objectives and implementation plans of CICE. Second, it would tend to confuse the USTRANSCOM user community and tend to diffuse any efforts by the CICE Functional Committee to harness the enthusiastic participation of the users in the development of the new information-centric capabilities. Third, it would certainly significantly increase the complexity of an already very complex technical coordination task. And finally, it would divert scarce resources to information systems projects that are not necessarily aligned with USTRANSCOM's principal objectives and strategies for transitioning to a knowledge management environment.

It is therefore recommended that concurrently with the short-term implementation strategies outlined in the previous section (see Section 5.2), plans be developed for the contractual realignment of at least one of the existing DoD transportation contracts (e.g., GTN-21) to serve as the initial primary vehicle for the implementation of CICE.

5.4 Guiding Principles for the Implementation of CICE

The following recommended guiding principles for the implementation of CICE are not intended to be comprehensive nor cast in concrete. Governing guidelines of this kind need to be subject to continuous review to ensure that they reflect the dynamic situational changes that will influence the objectives, implementation strategies, and user priorities of any large IT program such as CICE. The recommendations are grouped under four categories: process; architecture; system behavior; and, information models. As far as possible, each category reflects and includes the "Best Practices and Principles" described in the Technical View (Section TV-2.3.1) and the "Universal Principles" described in the Systems View (Section SV-2.2) of the DTS Enterprise Architecture (USTRANSCOM 2003a).

Process:

(1) Assign the responsibility for defining and maintaining the integrity of business rules to specific custodians within each business unit.

(2) Select tools based on capabilities rather than one integrated tool that may blur the logical application boundaries.

(3) Insist on design, architecture, component, and software development process documentation.

Architecture:

(1) Design applications to be highly granular and loosely coupled. This is desirable not only at the physical level, but also at the interaction or collaborative level (e.g., agents need not be explicitly aware of each other in order to effectively collaborate). Loosely coupled architectures provide greater flexibility to deal with architectural and implementation changes, in addition to overall maintenance and greater potential for reuse. In a similar sense models of collaboration, where collaborators (e.g., agents, management modules, etc.) communicate indirectly with each other via shared ontology(s), support a more flexible and dynamic interaction in which collaborators can be added, modified, or even temporarily removed during execution.

(2) Support the notions of flexibility, scalability, and performance by considering the distribution of various aspects of the system architecture (e.g., ontology objects, logic tier, etc.). Monolithic systems have inherent issues in terms of scalability and performance under evolving system loads. The ability to run a resource-intensive component of the system (i.e., heavy weight agent, etc.) in a more resourced environment (e.g., a dedicated machine or processor), even if only

for a limited period of time, can be an effective strategy for increasing overall system performance and avoiding performance bottlenecks.

(3) Take advantage of the qualities inherent in object-oriented design, such as encapsulation, reuse, and extensibility. Object-oriented design and development principles are industry-proven methods for creating high quality and manageable software.

(4) Promote platform independence to provide additional flexibility in operating environments. In particular, avoid platform-specific and hard-coded interfaces. Programming languages such as JAVA provide effective means for avoiding the coupling of software to particular operating systems. The result of such independence is that the same piece of software can be executed on a variety of operating platforms without requiring any recoding or even reconfiguration.

(5) Refrain from prematurely allowing implementation constraints to drive architectural decisions. Although implementation limitations will ultimately play a role in the target system, considering implementation details too early in the design process may complicate the design process in addition to needlessly limiting architectural choices.

(6) Insist on multi-tier architectures that separate information, logic and presentation, and that are service-oriented. Such architectures tend to be scalable and lend themselves to future modification as business rules change. An example of such benefits can be seen when a new user-interface is required. Having an architecture in which there is a decisive split between ‘presentation’ and ‘logic’ makes such modification a significantly more manageable task than if both tiers are tightly integrated.

(7) Generalize the design of application interfaces so that these interfaces can accommodate a wide range of data and information exchanges. Among other techniques, this can be effectively achieved through employing various design patterns (e.g., Adapter, etc.) that essentially abstract the fundamentals of some behavior into an adaptable interface thus supporting multiple implementations.

(8) Implement business rules as discrete, executable components or services thus providing greater flexibility in configuration and improved manageability

System Behavior: (1) Aim at preserving bandwidth by minimizing the amount of communication to that which is needed. For example, in some cases it may be perfectly acceptable for notification of an event to not contain the value(s) of the event. In some cases, knowledge of the fact that the event occurred is sufficient for the receiving client to take appropriate action. Clients wishing to obtain more details of the event can initiate the appropriate interrogation as a follow-on activity. To achieve additional performance improvements a flexible architecture should support both valued event notification in addition to unvalued event notification.

(2) Take advantage of the opportunity of an application to perform background processing. For example in the case of agent analysis, as soon as information begins to become available agents can engage in their analysis as a background activity capitalizing on otherwise unused resources (e.g., during user interaction, etc.). This can be an effective means of increasing overall system performance even if such processing needs to be partially repeated as additional information becomes available.

(3) Promote the notion of accessing data through business rules. Since data is created and used by business processes it should be managed by the component of a system that has responsibility for the business process.

Information Models: (1) Strive for representational accuracy as opposed to 'skewing' an available definition to fit a particular need. This is the 'other side of the coin' when promoting reuse. While it is certainly advantageous to reuse existing, well designed models and architectural fragments it is important to insure that the pattern does in fact effectively suite the problem and does not result in a poor adaptation.

(2) Represent applicable constraints in the ontology in preference to other system components, such as agents. Relationships and abstraction are valuable techniques in this regard. Encapsulating information and knowledge-level constraints into their native representation (ontology) both simplifies the other components (e.g.,

agents, etc.) in addition to strengthening the relationship between the information or knowledge and their constraints.

(3) Represent fundamental concepts and notions at the higher levels of the abstraction tree, to take advantage of inheritance. For example, applying a wide range of entities and concepts, the generalized notion of *trackability* would typically be represented at a higher level in the inheritance tree since it is a fundamental concept that is likely to apply to a large number of entities.

(4) Start with basic concepts and descriptions in skeletal form, making considerable use of relationships, when developing a model. This ensures that the details of specific characteristics do not impair the formulation of the basic structure.

(5) Take advantage of opportunities to reuse existing models. At the very least, such models may form a convenient basis from which to derive a more accurate model to suite the actual needs.

(6) Allow ontology development, similar to system behavior, to be driven by the users, utilizing use-cases as a tool. Consider what questions will be asked of the model. In most cases the fundamental purpose of such models is to answer various questions.

(7) Always allow for extensions. Abstraction is a powerful aid in this endeavor. Similar to the use of architectural patterns, adopting analysis patterns that lend themselves to extensibility and adaptability can provide an effective means of representing extended concepts and descriptions in the future.

(8) Avoid the tendency to dilute or wholly omit biases in favor of a model that only partially represents its user's perspectives. Perspective is both an inherent and highly useful aspect of accurately representing a particular domain. Employing the notion of Perspective Filters can be of significant assistance in this area.

(9) Subdivide a potentially large domain into smaller more manageable, and potentially reusable, sub-domains.

CDM Technologies, Inc., 2975 McMillan Avenue, San Luis Obispo, CA 93401 - TEL: 805-541-3750; FAX: 805-541-8296

(This page is intentionally left blank.)

6. References, Bibliography, and Acronyms

6.1 References

Biermann A. and J. Feldman (1972); 'A Survey of Results in Grammatical Inference'; Academic Press, New York.

Boehm B. (1986); 'A Spiral Model for Software Development and Enhancement'; ACM SIGSOFT Software Engineering Notes, 11(4), AC, New York, New York (pp.14-24).

Buschmann F, D Schmidt, H Rohnert, and M Stal (1996); 'Pattern-Oriented Software Architecture: A System of Patterns'; Vols. 1 and 2, John Wiley and Sons, New York, New York.

Chandrasekaran, B., Josephson, John R. and Benjamins, V. Richard (1999); 'What are Ontologies, and Why Do We Need Them?'; IEEE Intelligent Systems, 14(1), January/February.

CJCS (2000); 'Joint Vision 2020'; Joint Chiefs of Staff, Director for Strategic Plans and Policy (J5), Strategy Division, US Government Printing Office, Washington DC, June 2000.

Cockburn A. (2001); 'Writing Effective Use Cases'; Addison-Wesley, Reading, Massachusetts.

Cohen B. L. and C. A. Sammut (1978); 'Pattern Recognition and Learning With a Structural Description Language'; Proceedings Fourth International Joint Conference on Artificial Intelligence, IJCPR, Kyoto, Japan (pp.394).

Covey S. (1989); 'The 7 Habits of Highly Effective People'; Fireside, Simon and Schuster, New York, New York.

Drucker P. (1993); 'Post-Capitalist Society'; Harper Business, New York, New York.

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

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

Gilb T. (1988); 'Principles of Software Engineering Management'; Addison-Wesley, Reading, Massachusetts.

Helgesen S. (1995); 'The Web of Inclusion'; Bantam Doubleday Dell Publishing Group, New York, New York.

Kennedy J. and R. C. Eberhart (2001); 'Swarm Intelligence'; Morgan Kaufmann, San Francisco, California.

Koza J. (1992); 'Genetic Programming'; MIT Press, Cambridge, Massachusetts.

McClelland J. L. and D. E. Rumelhart (1988); 'Explorations in Parallel Distributed Processing: A Handbook of Models, Programs and Exercises'; MIT Press, Cambridge, Massachusetts.

Martin J. (1991); 'Rapid Application Development'; Macmillan, New York, New York.

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

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.

OSD (1997); 'C4ISR Architecture Framework'; Version 2.0, Department of Defense (DoD), December 18, 1997 [www.c3i.osd.mil/org/cio/i3/AWG_Digital_Library/index.htm]

OSD (1999); 'Department of Defense (DoD) Information Management (IM) Plan'; Version 2.0, October 1999 [www.c3i.osd.mil/org/cio/ciolinks/references/itmstpln/itmstpln-memo.html]

OSD (2001); 'Global Information Grid (GIG) Capstone Requirements Document'; JROCM 134-01, August 30, 2001 [www.dsc.osd.mil/gig/GIG_Arch_v1.0_(Final)/GIG_CRD_(Final).pdf]

OSD (2003); 'Global Information Grid (GIG) Architecture'; current version, Department of Defense (DoD), Chief Information Officer, Architecture & Interoperability Directorate, US Government Printing Office, Washington DC.

Palmer S. and J. Felsing (2002); 'A Practical Guide to Feature-Driven Development'; Prentice Hall, Upper Saddle River, New Jersey.

Pohl J.; 'The Emerging Knowledge Management Paradigm: Some Organizational and Technical Issues'; InterSymp-2003, Focus Symposium on Collaborative Decision-Support Systems, Baden-Baden, Germany, July 28 to August 1, 2003 (Preconference Proceedings, June 2003, pp.11-26).

Pohl J. (2001); 'The Meaning of an Information-Centric Computer Environment'; InterSymp-2001, 13th International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, July 30 – August 3.

Pohl K. (2001); 'Perspective Filters as a Means for Interoperability Among Information-Centric Decision-Support Systems'; 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, June 5-7, 2001.

Stapleton J. (1997); 'DSDM: Dynamic Systems Development Method'; Addison-Wesley, Reading, Massachusetts.

Stenbit J. (2003); 'Department of Defense (DoD) Net-Centric Data Strategy'; US Department of Defense, Chief Information Officer (CIO), Washington, DC, May 9, 2003.

Winston P. H. (1970); 'Learning Structural Descriptions from Examples'; Technical Report AI-TR-231, MIT, Cambridge, Massachusetts, September.

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

USTRANSCOM (2003a); 'Defense Transportation System Enterprise Architecture'; USTRANSCOM TCJ6-A, Scott Air Force Base, Illinois, 31 January 2003.

USTRANSCOM (2003b); 'Data Management Handbook'; USTRANSCOM Corporate Data Office (CDO), Scott Air Force Base, Illinois, August 2003.

6.2 Bibliography

Knowledge Management

O'Dell C., S. Elliott, and C. Hubert (2000); 'Knowledge Management: A Guide for Your Journey to Best-Practice Processes'; APQC's Passport to Success Series, American Productivity and Quality Center, Houston, Texas.

O'Dell C., F. Hasanali, C. Hubert, K. Lopez, and C. Raybourn (2000); 'Stages of Implementation: A Guide for Your Journey to Best-Practice Processes'; APQC's Passport to Success Series, American Productivity and Quality Center, Houston, Texas.

O'Leary D. and P. Selfridge (1999); 'Knowledge Management for Best Practices'; Intelligence, Winter (pp.12-23).

Pohl J. (2001); 'Transition from Data to Information'; InterSymp-2001, 13th International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, July 30 – August 3, 2001.

Pohl J. (2003); 'The Emerging Management Paradigm: Some Organizational and Technical Issues; InterSymp-2003, 15th International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, July 29 – August 1, 2003.

Smith G. and A. Farquhar (2000); 'The Road Ahead for Knowledge Management'; AI Magazine, Winter 2000 (pp.17-40).

Interoperability

Chaudri V., A. Farquhar, R. Fikes, P. Karp, and J. Rice (1998); 'OKBC: A Programmatic Foundation for Knowledge Base Interoperability'; Fifth National Conference on Artificial Intelligence, Menlo Park, California, AAAI Proceedings (pp.600-607).

Dawson R. (2000); 'Developing Knowledge-Based Client Relationships: The Future of Professional Services'; Butterworth-Heinemann, Boston, Massachusetts.

Dixon M. (2000); 'Common Knowledge: How Companies Thrive by Sharing What They Know'; Harvard Business School Press, Boston, Massachusetts.

Pohl J. (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.

Ontology-Based Software

Ambler S. (2002); 'Agile Modeling: Effective Practices for Extreme Programming and the Unified Process'; Wiley, New York, NY, 2002.

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

Chandrasekaran B., J. R. Josephson, and V. R. Benjamins (1999); 'What are Ontologies, and Why Do We Need Them? IEEE Intelligent Systems, 14(1), January/February.

Cunningham W. and R. Johnson (1997); 'Analysis Patterns: Reusable Object Models'; Addison-Wesley, Reading, MA, 1997.

Farquhar A, R. Fikes and J. Rice (1997); 'The Ontolingua Server: A Tool for Collaborative Ontology Construction'; International Journal for Human-Computer Studies, 46(6), (pp.707-727).

Fowler M. (2003); 'Patterns of Enterprise Application Architecture'; Addison-Wesley, Reading, MA, 2003.

Fridman-Noy N. and C. D. Hafner (1997); 'The State of the Art in Ontology Design: A Survey and Comparative Review'; AI Magazine, 18(3), Fall.

Kruchten P. (2000); 'The Rational Unified Process: An Introduction'; Addison-Wesley, Reading, MA, 2000.

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

Leighton R. (2000); 'Information Representation Basis of Decision Support Systems'; Office of Naval Research, Workshop on Collaborative Decision-Support Systems, Embassy Suites Hotel, San Luis Obispo, CA, May 2-4, 2000.

Noy N. and C. Hafner (1997); 'The State of the Art in Ontology Design: A Survey and Comparative Review'; AI Magazine, Fall 1997 (pp.53-74).

Noy N. and D. McGuinness (2001); 'Ontology Development 101: A Guide to Creating Your First Ontology'; Stanford University, Stanford, California [www.smi.Stanford.edu]

Pohl K. (2001); 'Perspective Filters as a Means for Interoperability Among Information-Centric Decision-Support Systems'; Office of Naval Research, Workshop on Collaborative Decision-Support Systems, Quantico, VA, June 5-7, 2001.

Rumbaugh J., M. Blaha, W. Premerlani, F. Eddy and W. Lorensen (1991); 'Object-Oriented Modeling and Design'; Prentice Hall, Englewood Cliffs, NJ, 1991.

Taylor D. (1990); 'Object-Oriented Technology: A Manager's Guide'; Addison-Wesley, Reading MA, 1990.

Uschold, M. and M. Gruninger (1996); 'Ontologies: Principles, Methods and Applications'; Knowledge Engineering Review, 11(2), June.

Uschold, M. and M. King (1995); 'Towards a Methodology for Building Ontologies'; Workshop on Basic Ontological Issues in Knowledge Sharing held in conjunction with IJCAI-95.

Warmer J. and A. Kleppe (1999); 'The Object Constraint Language: Precise Modeling with UML'; Addison-Wesley, Reading, MA, 1999.

Zang M. (2003); 'Data, Information, and Knowledge in the Context of SILS'; Office of Naval Research, Workshop on Collaborative Decision-Support Systems, Quantico, VA, Sep.18-19, 2003.

Zang M. (2003); 'The Knowledge Level Approach to Intelligent Information System Design'; InterSymp-2003, 15th International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, July 29 – August 1, 2003.

6.3 Acronyms

AIT	Automatic Identification Technology
AMC	Air Mobility Command
AMS	Acquisition Management System, or, Asset Management System
API	Application Programming Interface
AT-21	Agile Transportation for the 21st Century
ATMS	Assemble Tracking Management System
BDSS	Business Decision Support System
CAMS-FM/G081	Core Automated Maintenance System for Mobility
CDE	Corporate Data Environment
CDO	Corporate Data Office
CDS	Corporate Data Solution
CG	Commanding General
CICE	Corporate Information-Centric Environment
CIM	Corporate Information Management
COI	Communities of Interest
CONUS	Continental United States
CORBA	Common Object Request Broker Architecture
CRIS	Corporate Resource Information Source
DAAS	Defense Automatic Addressing System
DDDS	Department of Defense Data Dictionary System
DISA	Defense Information Systems Agency
DoD	Department of Defense
DTS	Defense Transportation System
EA	Enterprise Architecture
EBXML	Electronic Business XML
EDI	Electronic Data Exchange
ETL	Extract, Transform, and Load
GES	GTN Exercise Support
GIG	Global Information Grid
GTN	Global Transportation Network
GTN 21	Global Transportation Network 21

HQTC	Head Quarters USTRANSCOM
HTTP	HyperText Transfer Protocol
IBM	International Business Machines Corporation
IBS	Integrated Booking System
ICODES	Integrated Computerized Deployment System
IDEF	Integration Definition for Function Modeling
IT	Information Technology
ITO	Installation Transportation Officer
JMOC	Joint Mobility Operations Center
JOPES	Joint Operation Planning and Execution System
JTCC	Joint Transportation Corporate Information Management (CIM) Center
KM	Knowledge Management
LDM	Logical Data Model
MM	Master Model (formerly: Transportation Logical Data Model)
MSC	Military Sealift Command
MTMC	Military Traffic Management Command
ODS	Operational Data Store
OLAP	Online Analytical Processing
OV	Operational View
OSD	Office of the Secretary of Defense
SOAP	Simple Object Access Protocol
SV	Systems View
TCC	Transportation Component Command
TLDM	Transportation Logical Data Model
TPFDD	Time Phase Force Deployment Data
TV	Technical View
UDDI	Universal Description, Discovery and Integration
USTRANSCOM	United States Transportation Command
WPS	Worldwide Port System
XML	Extensible Markup Language

7. Keyword Index

X

Z

CDM Technologies, Inc., 2975 McMillan Avenue, San Luis Obispo, CA 93401 - TEL: 805-541-3750; FAX: 805-541-8296

(This page is intentionally left blank.)