ANALYZING C4ISR ARCHITECTURES THROUGH AN AUTOMATED DATA VISUALIZATION ENVIRONMENT

Raymond J. Curts, Ph.D., (CDR, USN Ret.)
Strategic Consulting, Inc.
Fairfax Station, Virginia
rcurts@ispwest.com
(703) 731-0301 (cell)

Douglas E. Campbell, Ph.D., (LCDR, USNR-R, Ret.)
Syneca Research Group, Inc.
Washington, D.C.
dcamp@syneca.com
(703) 627-4257 (cell)

ANALYZING C4ISR ARCHITECTURES THROUGH AN AUTOMATED DATA VISUALIZATION ENVIRONMENT

Raymond J. Curts, Ph.D., (CDR, USN Ret.)
Strategic Consulting, Inc.
Fairfax Station, Virginia
rcurts@ispwest.com
(703) 731-0301 (cell)

Douglas E. Campbell, Ph.D., (LCDR, USNR-R, Ret.)
Syneca Research Group, Inc.
Washington, D.C.
dcamp@syneca.com
(703) 627-4257 (cell)

ABSTRACT

Command and Control (C2) organizations continue to deal with the problem of architecture design, implementation and integration because they still must integrate multiple, interconnected, component "architectures" into joint, entity-wide information infrastructures. Experience has shown that most C2 decision-making continues to be based upon an infrastructure that consists of "stove-piped," legacy systems. Even when new architectures are designed and systems built to speed up the decision-making process, the procurement cycle, typically 10 to 15 years, lags significantly behind the current 12 to 24 month technology advancement cycle in IT related areas. In other words, an architecture is written, assessed, and published with a review and update every couple of years thereafter but never fast enough nor complete enough to keep up with technology nor operational needs. The authors continue to see this process as ill-defined, disjointed, disorganized, and failing to adequately complete even the first phase of architecture definition -- requirements collection and analysis. Indeed, the sheer mass of research, information and analyses required prevents it from being a fluid dynamic. In this paper, the authors will attribute this to the lack of consistent taxonomies, processes, organizational constructs and automated architecture assessment tools, and offer a few suggestions toward a solution.

INTRODUCTION. Complex information systems built for decision-making are commonplace in C2 operations. The ability to build, operate and maintain such systems is crucial to the effectiveness of C2. The authors' argument is that we need the ability to establish a solid information infrastructure for C2 decision-making based upon rigorous, standardized architecture definition, development, analysis, description and acquisition planning. Considering the vast quantities of information required, this can only realistically be achieved with automated support. Intelligent systems that support human judgment and choice tend to embody computer-implemented analytical and heuristic procedures that seek to combine knowledge about a domain (e.g., problem or situation) with methods of identifying, structuring, and reasoning about such a

domain. They incorporate, additionally, formal methods of reasoning about the domain that we must bring to bear when tasks are poorly understood, are ill-structured, or when the information available is incomplete, fragmented, or imperfect.
This paper describes the authors' attempts, through independent research and development, to create an automated toolset that deals with the processes of defining, manipulating, analyzing, assessing and selecting architectures and architectural components. The project was called the Data Analysis and Visualization Environment or DAVE®. Although the original implementation focuses on military systems, this discussion is sufficiently generic to be applied to any architecture that can be described by data elements. And, the authors contend, any and all architectures should be thus defined. Architectures have become far too large and complex to be handled in the traditional way with just textural documents and line drawings. Clearly, in this age of automation, better methods are available.

We are getting much better at establishing and managing C2 organizations and operations, but we continue to have difficulty assigning such widespread, global responsibility to any single organization. We must be careful to resist the urge to allow every C2 organization to design, develop and maintain its own separate architectural data structures for their decision-making capabilities. This is precisely the problem we are having today. Many organizations have a need to collect and maintain the data pertinent to their particular situation but we must be careful to prescribe some minimal set of standards so that they can all be seamlessly integrated into a single C2 architecture. Through the use of an automated toolset such as DAVE®, the authors are creating a data steward that is needed for the accuracy and completeness of a particular class of data (possibly by service or functional organization) that will ensure that data elements, format, schema, etc., are standard across all architectural data. This paper will give the reader a basic understanding of how a C2 decision support toolset such as DAVE® could best accomplish this effort.

BACKGROUND. This research originated in the late 1980s as a corporate Independent Research and Development (IR&D) project. In the early 1990s the U.S. Navy's Warfare Systems Architecture and Engineering (WSA&E) program at the Space and Naval Warfare Systems Command (SPAWAR) was struggling with the integration, analysis and interpretation of more than 59 independent, paper-based (i.e., textual) architectures that interconnected both physically and functionally. The use of DAVE® to support this effort was an attempt to add rigor and repeatability while shortening the architecture life cycle. At the time, the Navy was attempting to fold these architectures together like chapters in a book. The process was on a two year cycle; i.e., all architectures were supposed to have been written, assessed and published, with a review and update, every two years thereafter. Unfortunately, even thought DAVE® was used in small pockets within the overall architecture development process, the cycle took longer than two years to complete and, while not totally abandoned, the effort was significantly reduced. Later, DAVE® was again used to support analysis of Unmanned Aerial Vehicle (UAV) payloads under another Navy contract. Recently, additional independent research by the authors has led to a more refined paradigm based on Object-Oriented (OO) concepts.

Although the original use of DAVE® was focused on supporting military Command and Control (C2), Electronic Warfare (EW) and other such Information System architectures, the concepts are generic and can be applied to any architecture that can be described by data elements. And,

as the authors contend, any and all architectures should be thus defined. Architectures have become far too large and complex to be handled in the traditional way with just textural documents and line drawings. Clearly, in this age of automation, better methods are available.

Today the C2 community and the IT community in general have a similar pressing need for the exploratory development of their automated information technology (IT) system architectures. C2 strategies and IT system acquisitions are undergoing deliberate changes in many large organizations, be they military, civil government or commercial. All have stand-alone, "stove-piped" legacy systems, interoperability issues, cost and security issues giving rise to these deliberations. Over the last few years the authors have examined several approaches, most recently a top-down approach to R&D and acquisition decisions, guided by high level requirements. What we find in nearly every case is an ill-defined process that never seems to form a solid foundation – by defining the architecture through requirements collection and analysis. Indeed, the shear mass of research, information and analyses required usually prevent an IT system from becoming a fluid dynamic. Much of this can be attributed to the lack of convenient, consistent taxonomies, processes, organizational constructs and tools [Curts, 1989a], [Curts, 1989b], [Curts, 1999]. The use of DAVE® to support these efforts helps to focus on compliance with these requirements. Rigorous mission analysis, organizational level perspectives, and systems architectures become DAVE®'s end products which thus allow well defined, coordinated, smooth and effective upgrades / transitions.

Currently, many large organizations perform this architectural development manually. The process is iterative and involves many steps and coordination among large, diverse groups. The current method is not only cumbersome; it is also full of inconsistencies. Add to this the fact that the accepted format for architecture products (hardcopy text and line drawings) is neither conducive nor responsive to either analysis or change. The significance of this problem with respect to the current process is that the results of the architecture formation have a major impact on the composition of C2 infrastructures and the interoperability of their component systems. Therefore, due to the critical nature of C2, the need for an architecture process, and the rapid advancements in IT technology, tools are needed to enhance the focus on areas where resources and expertise can help develop an efficient IT architecture that is responsive to C2 organizational needs and the technology changes that occur in today's world. In the authors' opinions, the DAVE® research project is just about the only, if not *the* only, toolset of its kind directly focused on C2 organizational needs of their IT systems based on architecture definition, development, analysis, description and acquisition planning.

Benefits from Automation. The full-scale development of the tools and databases designed in this research effort provide a mechanism to create architectures in an integrated, data format that constitutes a one-time creation effort with only maintenance and updates needed for future architectures, as opposed to the constant re-creation of multiple architecture documents. Automation will save time, money, and personnel resources while adding rigor and repeatability.

Automation would also allow a shift in resources with less time and energy spent on developing the functional and physical architecture definitions and more time spent on architecture analysis, assessment, option development and acquisition planning where the real benefits lie. There is also a higher degree of consistency in both the database representation and analytical results. In

addition, it provides the capability to make use of the large amounts of data that already exist in various data stores.

Finally, with the capability to isolate which architecture elements are contributing most to the deficits in capability (through discrimination analysis), and experiment with increasing the capability in some areas to make up the deficit (through sensitivity analysis, what-if analysis and options development), the organization is able to increase IT's capabilities in the most cost effective manner. In other words, with the availability of funds always limited and decreasing, one of the greatest benefits of implementing this system is the capability to assist analysts in determining which acquisition strategies (for systems, platforms, sites and so forth) should continue in order to maximize return on investment, by maximizing capability.

DAVE® – WHAT IS IT AND WHERE ARE WE TODAY? The initial effort for the automation of the IT systems architecture development process in the late 1980s was to develop innovative system architecture tools. The goal of the prototype system was to automate a portion of the architecture process using a generic, artificially generated, sample data set. The tool had such promise that it was cloned to support U.S. Navy architecture studies. It successfully demonstrated how a fully operational toolset could enhance the performance of the architecture process, how it could be used to support budget decisions, and how such a tool could increase the fidelity ("goodness") of a resulting architecture. As a result of this effort, there were three main objectives achieved in support of the primary goal. The first was to design the tools (models and databases) which would aid in the automation of the architecture process. The second was to capture the processes within a C2 environment and establish these processes in a way that could be manipulated through automation. The third objective was to then develop a working prototype system that automates a portion of that design and the identified C2 processes. This prototype, dubbed DAVE® for Data Analysis and Visualization Environment, not only served as a proof-of-concept for the follow-on goal of automating the entire process, but also served as a model for a full-scale system design.

Prototype System Overview. The prototype system for automating the architecture process was composed of a database and several analytical modules that analyzed the data individually as well as the relationships among data elements. A high level architecture view of this system is provided in Figure 1. This particular depiction originated with Andrew P. Sage as a generic high-level architecture for any Decision Support System (DSS). From the bottom up, Dr. Sage referred to the layers as the Database Management System (DBMS), the Model Base Management System (MBMS) and the Dialogue Generation and Management System (DGMS) respectively [Sage, 1991].

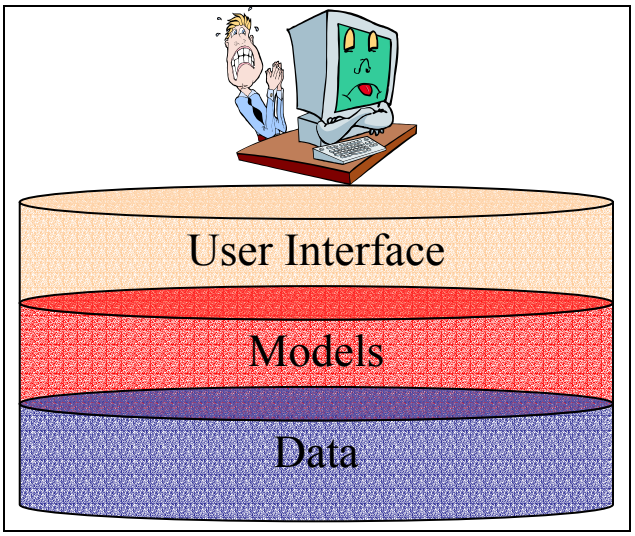

Figure 1. Decision Support System (DSS).

The Database Management System (DBMS). The DAVE® database schema was conceived as an integrated set of data tables representing system functionality, risk, threat and vulnerability data in terms of functional requirements and capabilities. The initial data structure was implemented in a set of flat files that could be easily imported into the tool. Although it is important to design a single, overarching data schema, there are multiple separate data sets with which the architecture process must interact, the most basic of which are functional requirements of the goal, "ideal" system and existing capabilities of the current, physical architecture. In addition, the data must contain information on system vulnerabilities and potential threats. First, a generic design for the database schema was created. The designs for these data sets are conceptually the same, the fundamental difference among them being in the data they contain. For example, the characteristics of the existing Physical (that is, own organization) and Threat (i.e., potential adversaries) tables are identical except that the equipment list (or rows / instantiation) for the Physical database contains data pertinent to the owner or those considered part of the organization, and that for the Threat database contains hostile organization data (e.g., hackers, direct competitors or any other organization from whom we might expect unfriendly or competitive action).

It became obvious from the very beginning that instantiating the current descriptions (mostly text and line drawings) of physical and functional "things" would make comparison very difficult. A single unifying structure was needed. After many iterations and testing of potential schema, the authors determined that a functional description would be best overall and a relational data schema was designed to capture the appropriate data. As we shall see later, that schema has recently been re-cast in the Object-Oriented (OO) paradigm.

One of the most important characteristics of the database design is the capability to select a variety of dimensions (or views) of the data. The analyses need to retrieve data elements with respect to time frames (e.g., Baseline or Current and Future or Desired/Goal), as well as selecting relationships among elements which may or may not be in the same data set (that is, selecting a cross section of functions, systems, suites, sites, etc.). The architectural data was initially

categorized into four data sets containing functional, physical, adversary, and technological architecture data as follows.

- **Functional Architecture Database.** The Functional Architecture Database is composed of the required functionality that must be performed by the physical architecture throughout the appropriate phases or timeline for the architecture in question. It consists of hierarchically nested functions that must be performed, in essence, a very detailed functional specification of the desired, goal or notional system.

- **Physical Architecture Database.** The Physical Architecture Database contains the existing hardware components, such as sites, suites, sensors, IT systems, and communication links that currently fulfill the functional needs of the organization. These components have an explicit hierarchical relationship as well as relationships to other physical and functional database elements and are also described functionally. I.e., A suite consists of some set of systems that are described by the total set of functional capabilities that they provide. For design purposes, space allocation, and other obvious reasons, additional information on physical systems must be kept such as location, size, weight, etc. Therefore the database implementation must allow specification of these relationships. In addition to the functional requirements and physical capabilities architectures described above, any number of other architectural data sets could be developed for analysis. For example:

  - ***Adversary Architecture.*** The Adversary Architecture data set is the current physical architecture of the threat (e.g., hackers) or opponent. This database would be used in the assessment of how well the physical architecture performs the required functions against the threat (e.g., How well does the firewall prevent unwanted intrusions while allowing full access to authorized users? How effective are the anti-virus countermeasures?). The database elements would have a format virtually identical to the above physical architecture example.

  - ***Technological Architecture.*** This is a notional physical architecture composed of physical architecture components (that is, platforms, systems, and so forth) that appear technologically feasible in some specified future time frame. This database is a crucial part of the Options Development Step in the architecture process. As before, the systems are described functionally and the data elements have a format virtually identical to the above physical architecture example.

The Model Base Management System (MBMS). This is the assessment layer and it is at the very heart of the automation capability. The goal of the models was to provide an analyst with a determination of how well existing components accomplish stated functional requirements against, or in light of, a known environment, scenario or threat. Since this is the most critical component of the automated tools, we have provided a more detailed design in this section.

There are several modules which comprise the Assessment Suite: Discrimination, Sensitivity, Performance Evaluation, Explanation, Database Instantiator, and Shortfall/Overlap Analyzer. These are briefly described as follows:

- **Database Instantiator Module.** The Database Instantiator module maps the databases into hierarchies. In so doing, it selects the view of the database, relationships among the elements, and the attributes (or parameters) of objects which need to be represented. It is this module that allows the assessment algorithms to remain generic, because the objects are instantiated at run-time (that is they are not domain or database dependent).

- **Performance Evaluation Module.** This module calculates the performance values for each of the elements in the hierarchy. The calculation is built upon an innovative combination of a semantic network and the Multi-Attribute Utility Analysis (MAUA) evaluation paradigm. Each element's total score is based on an additive, weighted scoring of related elements in the hierarchy. There are a number of ways in which performance might be calculated based on certain conditions for the evaluation set in advance by the user, e.g., scenarios, missions, environmental conditions, and so forth. Add-on modules can be added to cover these relatively easily.

- **Explanation Module.** Very important to the overall functionality of the assessment is the analyst's capability to obtain detailed explanations of the results for both performance evaluations and for short-falls and overlap development. Analysts will not typically take the results of an automated assessment at face value (at least for critical analyses or during the first uses of the tool). Therefore, the system must be able to explain how certain conclusions were obtained. This explanation will take the form of an audit trail of the inference process and the rules used, in a form understandable by domain experts. There will also be a justification of the performance scores which were calculated in the form of discrimination analysis (*see* Discrimination Analysis module below).

- **Discrimination Analysis Module.** Another major feature of the initial prototype is the capability of performing Discrimination Analysis. Discrimination Analysis allows an analyst to view the contributing factors for a particular element's deficit from a perfect score. For each contributing factor, the magnitude of the contribution to deficit and the percentage contribution to the overall deficit is displayed. Examples of typical discrimination analysis displays from DAVE® are shown in Figures 2 and 3.

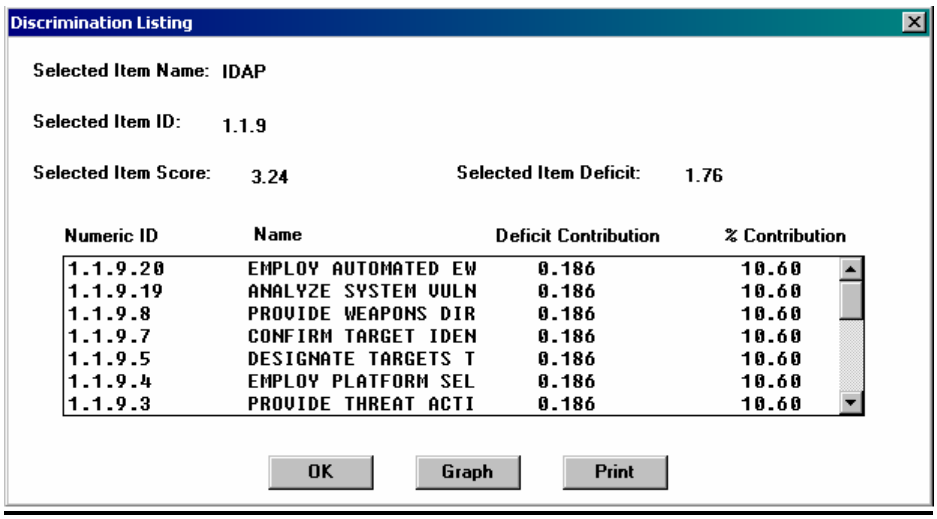

Figure 2. Discrimination Analysis Table.

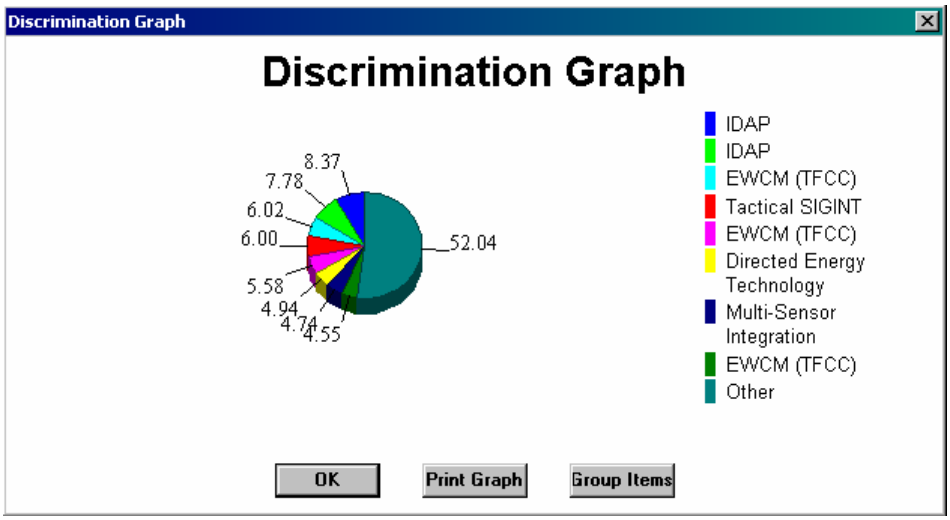

Figure 3. Discrimination Analysis Graph.

- **Sensitivity Analysis Module**. Sensitivity Analysis provides the capability to modify the weights or scores which are contributing to an entity's deficit. After the modifications are made, the analyst can then view the impact of these changes by re-calculating the scores and updating a color-coded display. In other words, the analyst can perform "What-if?" analyses to determine what minimal changes could be made to the current architecture to derive improved or even maximum benefit. While sensitivity and what-if analyses employ the same concepts and techniques, the authors view the two as distinctly separate methods. What-if analysis tests specific points in the design to determine the impact of change. Sensitivity analysis, on the other hand, focuses on determining which nodes are most susceptible to change (i.e., where can we achieve the largest benefit from the smallest amount of change?).

- **Shortfall / Overlap Assessment Module.** This module is designed to perform systematic comparisons of the systems' parametric data. While not included in the

initial prototype, this module is a high priority for implementation early in the next phase. Currently two types of assessments are intended. The first is an assessment within a particular data set with respect to other components of that same architecture. For example, synergistic components may have an interacting functionality such as the combined effect that jamming and chaff together have on target masking. Therefore, an assessment of the interactions between supporting and, possibly, antagonistic platforms and systems should be completed.

The second type of assessment allows a mapping between databases to identify short-falls and overlaps in functional capability. For example, mapping the Physical Architecture to the Functional Architecture would identify where the current Physical platforms and systems fail to meet a certain requirement (desired functionality) specified in the Functional Architecture, thus causing a shortfall. Conversely, an overlap or redundancy exists where there are several physical platforms and/or systems that meet a specific requirement. It is recognized that some overlap (i.e., redundancy) may be desirable, but excessive duplication is costly in a number of areas such as space, weight, and cost, and should generally be avoided. In another scenario we might compare the parameters of own organization hardware systems (potential vulnerabilities) with the counter hardware systems of potential threats to see how well they match up on certain key parameters thus identifying potential risks.

- **Option Development Module.** Option Development is where the possible Future Systems Architecture is created and analyzed, and it is closely tied to the analysis of shortfall and overlaps. This module is a tool for analysts to perform What-If and Cost-Benefit analyses for development of future architectures. The design of this module should also allow the clustering of functionalities when building the functional hierarchy. For example, certain functions typically have been grouped because they are so grouped on an existing platform; for example, Hard Kill, Classify, and Detect functions are typically grouped because they are typically performed by an aircraft, such as the F-14. It may be more appropriate to group functions that logically and technologically work together, and then identify a platform or system that would potentially meet these logically "clustered" functionalities. This kind of clustering can lead analysts to create new, innovative platforms / sites / systems.

The Dialogue Generation and Management System (DGMS). This is the Graphical User Interface (GUI) layer designed to give the analyst a graphical overview of the status of the architecture elements (e.g., platforms, systems, functions, and so forth). This is done by displaying the score of each element using colors: red, orange, yellow, yellow-green, or green, corresponding to a rating scale of one (1) through five (5), respectively where 1 is bad (red) and 5 is good (green). Each element's node is connected to related elements via arcs. Similar to Object Oriented (OO) concepts, these relationships are defined as parent / child relationship and generally refer to groupings, either physical or functional. In the Navy example, the ALQ-99 is modeled as a child of the EA-6B because it is an integral part of the EA-6B platform. Likewise for the functional databases, requirements are often grouped by the phases of battle (e.g.,

Electronic Intelligence (ELINT) is a child or component of the Surveillance / Reconnaissance Phase).

Along with the usual features of a typical GUI, other design features include the ability to define, select and display hierarchies in any order required plus the capability to isolate specified families (tracks) of nodes and to track them through the hierarchy as illustrated from DAVE® in Figure 4.

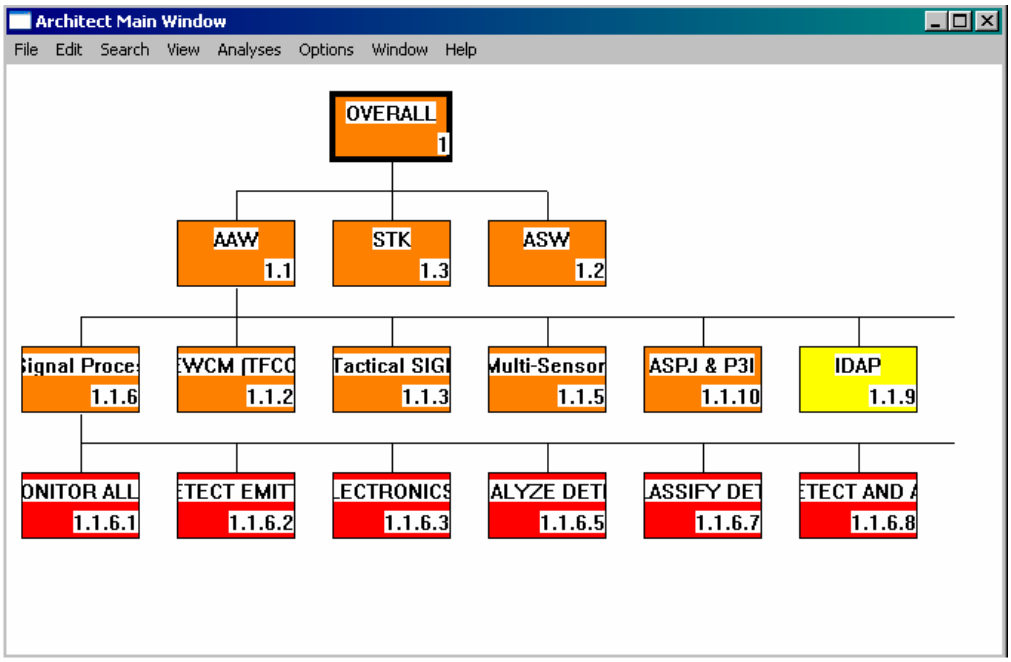

Figure 4. Functional Hierarchy.

Summary. As can be seen from the above, the designed system and databases simultaneously address two critical needs: 1) innovative concepts for future IT Systems; and 2) architecture automation tools for multi-site / organization / system architecture. Our methodology and the tools designed and prototyped in the initial phase are capable of assessing alternative (architecture) performance. Due to evolving technology, advancing rapidly in scope and capability, it is critical that we move forward to develop a set of modeling tools capable of assessing the integrated capabilities of systems / sites / segments / organizations while they are participating in combined, integrated operations. Today, far too much time is spent on constantly redefining architectures and producing the resultant documentation leaving little attention to analysis and option development where the real benefits lie (Figure 5).

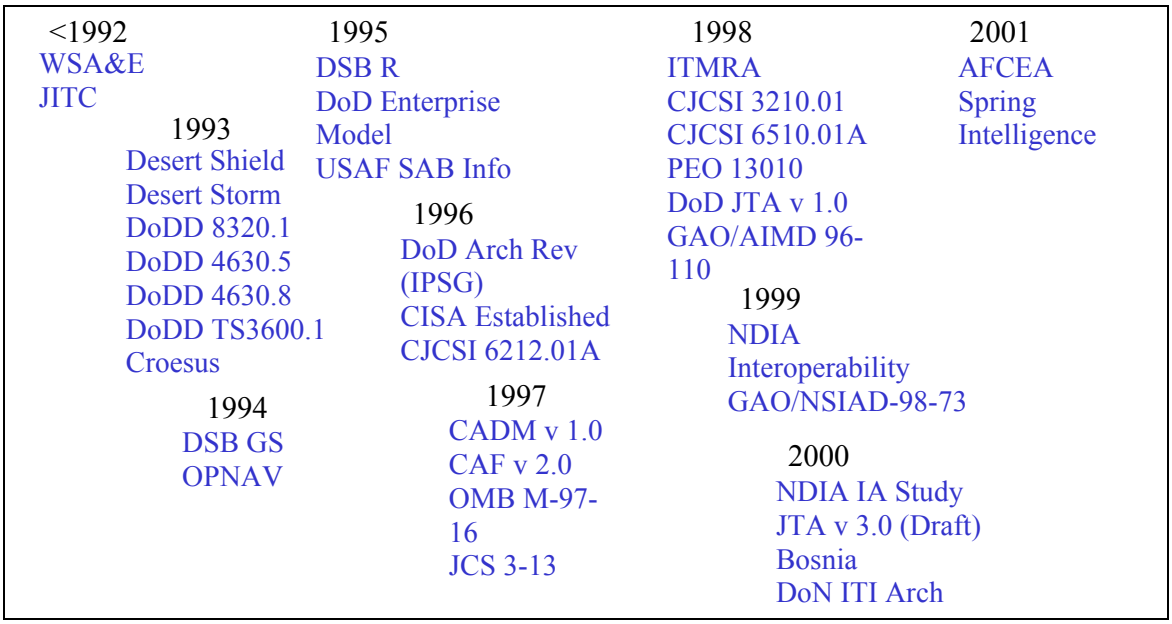

Figure 5. How Do We Get There? When Do We Get There?

THE OBJECT-ORIENTED PARADIGM & ARCHITECTURAL ATOMS

Object-Oriented Architecture. Architecture is a planning process or tool. The ultimate goal of any such process is, of course, to build a knowledge- and experience-base upon which to formulate decisions toward shaping the future. To this end, planners must be able to capture the necessary data to be able to organize and/or reorganize the components of a large, complex system so that it functions smoothly as an integrated whole. The decision-maker must be able to quickly manage, manipulate, and study the effort on a conceptual level so that it can be implemented on the physical level in some preplanned, logically structured fashion.

The ability to gracefully accommodate dynamic evolution, such as that needed in shortening the decision cycle, is an important research issue in database technology. Databases use several concepts, methodologies and programming paradigms to accurately document the "view of the world" and to assist the decision-maker in drawing conclusions from that data. In many cases, the conclusions arrived at by the decision-maker were not explicitly programmed into the model. In recent experience, these projects are more frequently being implemented in the form of object-oriented systems. This dynamic evolution begins just like the evolution of any living creature—from a set of common building blocks.

Building Blocks. As we have mentioned, there is an urgent need for a common description of architectures. Architectures are typically developed from functional requirements (in the case of notional architectures) or functional capabilities (in the case of physical, existing architectures) that we consider essential to the successful operation of our entity or achievement of our goals. These requirements have been expressed in many forms and at varying levels of granularity. However, if we adopt the concept of *functional requirements* as the building blocks of our architectural description, we have the opportunity to conduct direct comparisons of effectiveness, interoperability and a large variety of other descriptors that are of interest to us (Figure 6).

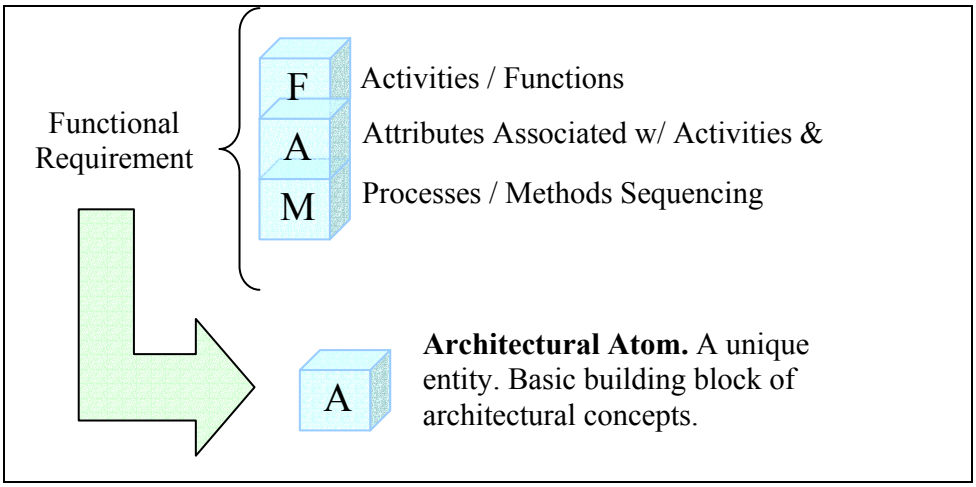

Figure 6. Architectural Atoms

Functional requirements lend themselves quite nicely to the object-oriented approach previously described. First, they can be represented as objects where each function or activity has a name, attributes are associated with that function, and processes, methods or activities are performed by the function. In this definition, Functional Requirement Objects become what the authors refer to as "Architectural Atoms," the building blocks from which any and all architecture components can be constructed. This structure allows us to take advantage of the other OO concepts. Here Architectural Atoms, the "leaf" nodes in the hierarchy, combine to form higher level objects while the unique attributes of high level objects can be inherited down the chain, Figure 7.

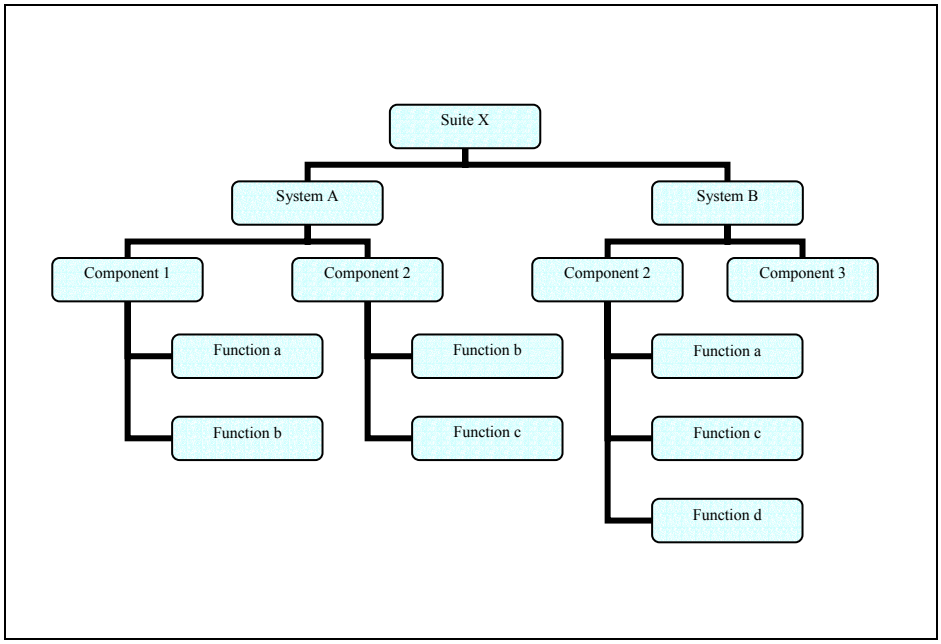

Figure 7. Functional Hierarchy.

Components then become systems and systems, in turn form a system of systems, or suite (Figure 8). From an architectural perspective these "Architecture Atoms" also allow us to readily

identify shortfalls (gaps in our functional capabilities) and functional redundancies (overlapping capabilities from multiple suites, systems or components) for further analysis. Shortfalls usually require attention while redundancies are often intentional and required in military C4ISR and other critical systems. Some redundancies, however, may be targeted for elimination in the name of efficiency and/or cost effectiveness.

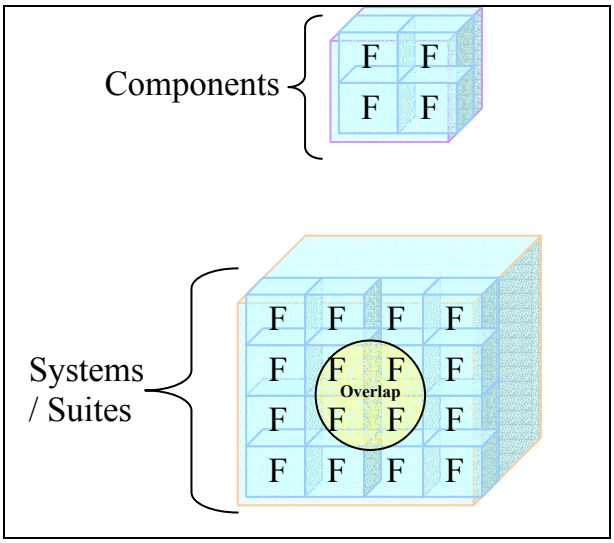

Figure 8. Functions and Components.

Thus, from a functional perspective, the entire architecture (functions, components, systems and platforms or suites, etc.) can be described using combinations of Functional Requirements (Figure 9).

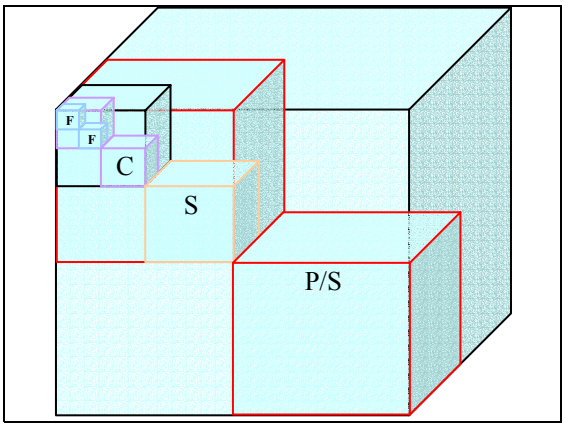

Figure 9. Functional Architecture.

Object-Oriented architectural components, when assembled, might resemble a Rubik's Cube (Figure 10). Each module represents a unique unit, system, or capability that can be combined with others in a virtually limitless number of ways. In addition to this physical flexibility, once assembled, the architecture can be viewed from multiple perspectives (also nearly limitless) to satisfy the requirements of the viewer.

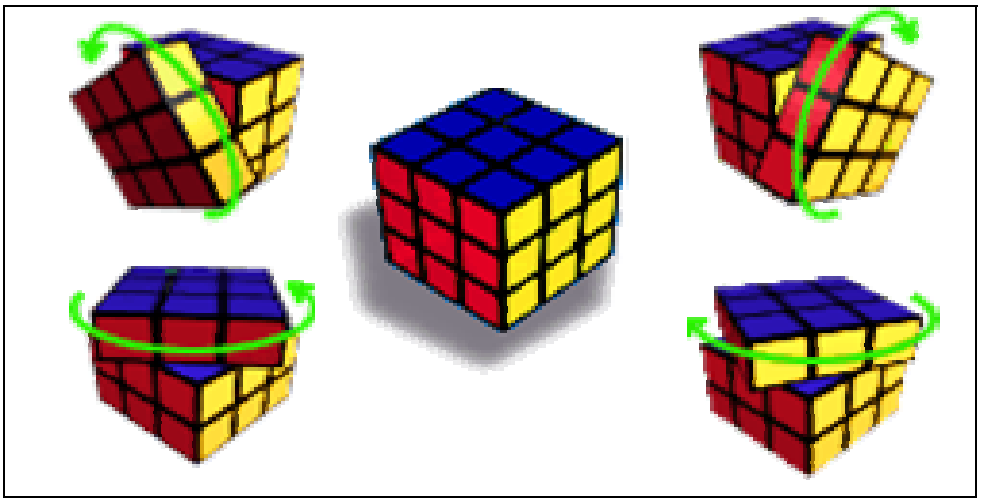

Figure 10. Rubik's Architecture Cube.

From the authors' perspective, this is one of the major disappointments with our current view of architectures and a primary reason that our systems are still not interoperable despite more than fifteen years identifying the issues. Numerous studies have shown that many useful architectures and architectural constructs exist. Unfortunately, they were all developed by different organizations, for different purposes, using similar but differing data, at varying levels of detail. Most were captured as documents (text and graphics) rather than as manipulable data. Though undoubtedly useful to the owners and developers of each, they cannot be directly combined nor compared in any meaningful way. Information Assurance (IA) has been a significant driver in Information Warfare (IW) circles recently. However, IA cannot be accomplished without interoperability, and we are not likely to achieve interoperability without a solid architectural foundation [Curts, 1999], [Curts, 2000].

Traditional database systems and visualization tools are limited in their data abstraction and representation power, and they fall short of providing important information management and data manipulation. The use of object-oriented data structures along with our concept of Architectural Atoms to support the decision-maker at various levels of abstraction is an important emergent concept where great strides can be made.

Advantages and Disadvantages. Object-oriented programming, and data management systems offer a number of important advantages over traditional control / data oriented techniques. For our purposes here, the most significant are:

- The modeling of all conceptual entities with a single concept, the object.
- The notion of a class hierarchy and inheritance of properties along the hierarchy.

Despite its many advantages, the object-oriented view is not perfect. However, though there are several drawbacks to OO systems, most are a direct result of their relative infancy and are expected to be resolved as the model matures. None are seen as a significant encumbrance to the concepts suggested here.

A more detailed discussion of advantages and disadvantages of the OO paradigm can be found in [Curts, 2001a], [Fowler, 1997], [Andrews, 1990], [Kim, 1990], [Manola, 1987], [King, 1986] and [Thomas, 1990].

SUMMARY. So, how does this concept of Object-Oriented Architecture help us achieve a better understanding of our architectural goals through a more robust, useful architecture definition? And, what will that do for the process of analyzing architectures, developing options and choosing acquisition strategies?

All things start with raw data elements, just as every living creature is composed of thousands of deoxyribonucleic acid (DNA) molecules. Whether strands of DNA or strings of "ones and zeros," each helps shape the individual and each controls the way that individual functions. In the case of architectures, that data equates to functional capabilities or functional requirements; in other words, our "Architectural Atoms." Alone they are not much more than well-structured data points. By combining these atoms into components we begin to build our architectural DNA and build the systems / capabilities hierarchy - collecting more and more information about our systems and situation. This leads, in turn, to a better understanding and awareness upon which to base our options development and acquisition decisions. In addition, it provides a common, base dataset that can be used by all systems so that all architectural views depict the same basic information (i.e., everyone operates from the same sheet of music). The simple action of standardizing, unifying and utilizing one common building block, coupled with a visualization tool such as DAVE® solves many of the problems that we have discussed here and many more that were only implied. In synopsis, we conclude the following:

1. Establish one standard method of representing and storing architectural data.
2. Collect all architectural data into a single central repository or a standardized, federated set of repositories to ensure that everyone is working with the same "big picture."
3. Ensure that the architectural data in standardized, common and available to all who need it. If everyone had ready access to the appropriate data in a common, useable form, we could make great strides toward solving the interoperability issue.
4. Allow for a more efficient visualization and analysis of capabilities across multiple services, battle forces, platforms, systems and organizations so that we can make more informed, efficient and effective acquisition decisions.
5. This higher understanding leads to a heightened level of awareness that allows us to see how the architecture fits from multiple views (Figure 10) thus enhancing option development and acquisition strategy selection.

Thus, by attacking and resolving the lowest level problem (the Architectural Atom), we can achieve a significant impact upon our warfighting capability while maximizing the bang from our acquisition buck.

Implementation of the paradigm described here is non-trivial. There are a number of hurdles to overcome:

1. Organizational – some, probably joint, centralized organization must be charged with collection, storage, retrieval, manipulation, comparison, maintenance and management of the data and / or data standards.
2. Technical – the data standards and database schema must be carefully designed to provide maximum flexibility and expandability.
3. Operational – the data must be readily available to whoever needs it.
4. Collection – very large quantities of data must be collected, verified, catalogued, sorted, stored and maintained.

Of these tasks, the first and last probably present the biggest challenges. Although we are getting much better at Joint organizations and operations, we continue to have significant difficulty assigning such widespread, global responsibility to any single organization. We must be careful to resist the urge to take the easy way out by allowing each service, agency or organization to design, develop and maintain its own separate architectural data definitions and data structures. This is precisely the problem that we have today. While a data steward should, no doubt, be assigned responsibility for the accuracy and completeness of a particular class of data (possibly by service or functional organization), it is important to ensure that data elements, format, schema, etc. are standard across all architectural data. This can most easily be accomplished in a single, centralized data repository, but a distributed or federated system will fulfill the same goal if properly implemented.

The biggest and most costly challenge will likely remain the collection of the huge mass of data required to drive such a system. However, if we consider the large amounts of money that are spent today by a very wide variety of offices collecting redundant and, often, inconsistent data, we might find that we will end up with significantly more information for less expenditure than we currently experience. Costs notwithstanding, what we have shown in this paper is that it is quite possible for an architect, acquisition agent or virtually any decision-maker to compare or otherwise manipulate large amounts of data in order to "observe, orient, decide and act" without suffering from information overload through an object-oriented "view of the world."

In addition there is the question of tools with which to visualize and manipulate the architecture. DAVE® was specifically designed as an architecture analysis and visualization tool, represents the perfect environment, and is ripe for upgrade to take advantage of these concepts.

FUTURE WORK. The potential utility of our exploratory research and development of an automated tool for the systems architecture process extends throughout the military and commercial systems domains. We have developed a generic capability to automatically define systems architectures based upon required functions (which derive from the anticipated operational environment) and available and anticipated technologies and existing physical systems (current embodiments of those technologies). One can imagine such a capability being readily extended to other types of architectures.

What we have goes beyond the typical Computer Aided Systems Engineering (CASE) tool to address analyses of entire architectures of systems. It presents the potential for an innovative exploitation in a societally critical domain of database management systems, rule-based expert systems, object-oriented programming, and conventional programming technologies.

DAVE® has proven useful in its current implementation but was intended as a proof of concept and suffers from age. With implementation of the concepts enumerated here, the next version could have significantly increased utility and flexibility. The following are the major upgrades currently planned for DAVE®:

1. Upgraded GUI – The graphic user interface was the state of the art at the time of its development in the early 1990's but has received no attention since. The current Windows environment has shown significant improvement in the years since DAVE® was first implemented.
2. Additional Analysis Tools – As described above, several analysis modules already exist in DAVE® but many others could be added. A simple Expert System for architecture comparison and analysis would likely be the first on the list [Curts, 1989b].
3. Upgrade to OODBMS – Migrating to the Architectural Atom and OO database concepts described above are probably the single most significant changes planned. Not only does this unify the data description but it allows more OO programming concepts to be utilized.
4. Reprogram in C++ - DAVE® was originally written in the C programming language. Since the first three things on this list require major changes anyway, this would be a good time to completely rewrite the software in C++ or another OO language that will provide better interaction with the OODBMS plus other benefits attendant with OO concepts.

REFERENCES

[Andrews, 1990] Andrews, Timothy, and Craig Harris. "Combining Language and Database Advances in an Object-Oriented Development Environment." Readings in Object-Oriented Database Systems, 186-196. Stanley B. Zdonik and David Maier, eds. San Mateo, CA: Morgan Kaufman, 1990.

[Coad, 1990] Coad, Peter, and Edward Yourdon. Object-Oriented Analysis. Englewood Cliffs, NJ: Yourdon Press, 1990.

[Coleman, 1994] Coleman, Derek, et. al. Object-Oriented Development: The Fusion Method. Englewood Cliffs, NJ: Prentice Hall, 1994.

[Cox, 1986] Cox, Brad J. Object Oriented Programming. Reading, MA: Addison-Wesley, 1986.

[Cox, 1987] Cox, Brad J. "Message/Object Programming: An Evolutionary Change in Programming Technology." Tutorial: Object-Oriented Computing, Volume I: Concepts, 150-161. Gerald E. Peterson, ed. Washington, DC: Computer Society Press, 1987.

[Curts, 1989a] Curts, Raymond J. A Systems Engineering Approach to Battle Force Architecture. Fairfax Station, VA: Strategic Consulting, Inc. (SCI), 1989.

[Curts, 1989b] Curts, Raymond J. An Expert System for the Assessment of Naval Force Architecture. Fairfax Station, VA: Strategic Consulting, Inc. (SCI), 1989.

[Curts, 1999] Curts, Raymond J., and Campbell, Douglas E. "Architecture: The Road to Interoperability." Paper presented at the 1999 Command & Control Research & Technology Symposium (CCRTS), U.S. Naval War College, Newport, RI, June 29 - July 1, 1999.

[Curts, 2000] Curts, Raymond J., and Campbell, Douglas E. "Naval Information Assurance Center (NIAC): An Approach Based on the Naval Aviation Safety Program Model." Paper presented at the 2000 Command & Control Research & Technology Symposium (CCRTS), U.S. Naval Postgraduate School, Monterey, CA, June 24 - 28, 2000.

[Curts, 2001a] Curts, Raymond J. and Douglas E. Campbell. Avoiding Information Overload Through the Understanding of OODA Loops, A Cognitive Hierarchy and Object-Oriented Analysis and Design. Annapolis, MD: C4ISR Cooperative Research Program (CCRP), 2001.

[Dittrich, 1986] Dittrich, Klaus R. "Object-Oriented Database Systems: The Notion and the Issue." International Workshop on Object-Oriented Database Systems. Washington, DC: Computer Society Press, 1986.

[Fowler, 1997] Fowler, Martin with Kendall Scott. UML Distilled: Applying the Standard Object Modeling Language. Reading, MA: Addison-Wesley, 1997.

[Kim, 1989] Kim, Won and Frederick H. Lochovsky, eds. Object-Oriented Concepts, Databases, and Applications. Reading, MA: Addison-Wesley, 1989.

[Kim, 1990] Kim, Kyung-Chang. "Query Processing in Object-Oriented Databases." Lecture Notes. Auston, TX: University of Texas, 1990.

[King, 1986] King, R. "A Database Management System Based on an Object-Oriented Model." Expert Database Systems: Proceedings of the 1st International Workshop. Larry Kerschberg, ed. Menlo Park, CA: Benjamin Cummings, 1986.

[Manola 1987] Manola, Frank A. "PDM: An Object-Oriented Data Model for PROBE". Cambridge, MA: Computer Corp. of America, 1987.

[Sage, 1991] Sage, Andrew P. Decision Support Systems Engineering. New York, NY: John Wiley & Sons, Inc., 1991.

[Taylor, 1997] Taylor, David A. Object Technology: A Manager's Guide, 2nd Ed. Reading, MA: Addison-Wesley, 1997.

[Thomas, 1990] Thomas, Agrawal, Jajodia and Kogan. "A Survey of Object-Oriented Database Technology." Fairfax, VA: George Mason University, 1990.

[Zaniolo, 1986] Zaniolo, Ait-Kaci, Beech, Cammarata, Kerschberg and Maier. "Object Oriented Database Systems and Knowledge Systems." Expert Database Systems: Proceedings from the 1st International Workshop. Larry Kerschberg, ed. Menlo Park, CA: Benjamin Cummings, 1986.

