Avoiding Information Overload Through the Understanding of OODA Loops, A Cognitive Hierarchy and Object-Oriented Analysis and Design

Dr. Raymond J. Curts, CDR, USN (Ret.)
Strategic Consulting, Inc.
5821 Hannora Lane
Fairfax Station, VA 22039-1428
(703) 395-9143
email: rcurts@erols.com

Dr. Douglas E. Campbell, LCDR, USNR-R (Ret.)
Syneca Research Group, Inc. (www.syneca.com)
P.O. Box 2381
Fairfax, VA 22031
voice/fax: (703) 876-0935
email: dcamp@aol.com

Abstract

At last year's Command & Control Research & Technology Symposium (CCRTS) at the Naval Postgraduate School in Monterey, CA, Dr. Richard E. Hayes from EBR and Dr. David T. Signori Jr. from RAND reported on their activities in the Information Superiority Metrics Working Group. Their presentation mentioned the OODA Loop (Observe, Orient, Decide, Act). In the late 1970s, Colonel John Boyd, USAF, wanted to understand why U.S. fighter pilots consistently won air combat engagements with their F-86 fighter aircraft in combat over Korea against pilots that flew Mig-15 aircraft with better maneuverability. His work came to be known as the OODA Loop or Boyd Cycle[1]. For these authors, the thought that a study on how fighter pilot decision-making in air-to-air combat during the Korean War had crept into a presentation on information superiority at the turn of the century was intriguing and suggested a closer look. What we discovered were logical stepping stones that led us from "fighter pilot decision-making" to previous discussions on "information overload" and from that to another CCRT presentation on the "cognitive hierarchy." The final stepping stone for this paper was the authors' abilities to mature this stepping stone process into a discussion of the methodology, applications and uses of Object-Oriented Analysis and Design (OOA&D) techniques to solve the problem of information overload. The authors have long been advocates of common shared data for the description and assessment of architectures, interoperability and information assurance. This paper continues that advocacy.

1. INTRODUCTION. The purpose of this research paper is to acknowledge what, if anything, can be done to assist the decision-maker to observe, orient, decide and act—that is, to achieve knowledge superiority or to avoid, or at least reduce, information overload. Decision-makers become "intelligent" enough to make decisions by the mental manipulation of data, information, or knowledge (levels of the cognitive hierarchy). We will show how a decision-maker would be able to compare or otherwise manipulate large amounts of data to "observe, orient, decide and act" without achieving information overload through an object-oriented "view of the world."

---

[1] Boyd, J. PATTERNS OF CONFLICT, December 1986. Unpublished study, 196 pages.

The dilemma of the decision-maker within the OODA loop is largely a problem of data collection, storage, retrieval, manipulation and comparison. Some sort of sophisticated set of algorithms, expert systems, or neural nets may eventually be required to fulfill the comparison portion of the decision-making process. For now, an object-oriented approach to data definition, storage, and manipulation would most certainly fulfill the decision-maker's need for a common dataset without resulting in information overload.

To begin, the use of an overall object-oriented Architecture is needed as a planning process or tool. The ultimate goal of any such process is, of course, to build the knowledge- and experience-base upon which to help decision-makers to better "observe, orient, decide and act." To this end, the authors conclude their research by acknowledging that planners must be able to organize and/or reorganize the components of a large, complex system so that it functions smoothly as an integrated whole. We open up this paper with a tribute to Col. John Boyd and his OODA Loop:

> *Machines don't fight wars. Terrain doesn't fight wars. Humans fight wars. You must get into the mind of humans. That's where the battles are won.*
>
> **Col. John Boyd**

2.   THE OODA LOOP. Common sense should tell us that a fighter aircraft with better maneuverability and similar speed characteristics should generally win the majority of "dog fight" engagements. However, this was not happening in actual air-to-air engagements during the Korean War. U.S. fighter pilots, despite flying aircraft with wider turn radii, were consistently beating adversary pilots and their aircraft. Based upon an in-depth study of the aircraft, Colonel John Boyd came to the conclusion that he was studying the wrong thing! It was not necessarily the characteristics of the aircraft that was the deciding factor in winning a "dog fight." Or, at least, not the only factor. It was the ability of the U.S. pilot to acquire the adversary first, and the speed with which the pilot's decision-making inputs reached the aircraft's control surfaces. Boyd's hypothesis was that a U.S. fighter pilot would win the "dog fight" because he could complete "loops" of decision-making quicker than his adversary. Boyd surmised that *quicker* was better than *faster*. Boyd's loop occurred in four distinct steps[2]:

- **Observe.** U.S. pilots could see their adversaries better and more completely because the cockpit design of U.S. aircraft ensured better visibility.
- **Orient.** Since our pilots acquired the adversary first, they could then react by orienting themselves toward the adversary faster.
- **Decide.** After reacting with their initial orientation, the U.S. pilot's level of training then allowed them, as a decision-maker, to act faster proceeding to the next combat maneuver.
- **Act.** With the next combat maneuver decided upon, U.S. pilots could then rapidly "input" aircraft control instructions, with the resultant faster initiation of a desired maneuver.

Based on these observations, Boyd's OODA Loop model of air-to-air combat was useful to the Air Force. His model also worked its way into the U.S. Marine Corps through the maneuver warfare writings of William F. Lind[3]. Lind, in his writings on ground combat and the role of maneuver in ground combat, re-oriented Boyd's OODA cycle and used it as a tool to describe how U.S. forces might be able to more efficiently prosecute ground combat. The OODA Loop thus became the method used to describe the process by which ground combat formations might be able to fight the enemy more efficiently by

---

[2] Ibid, page 5.

[3] Bateman, Robert L. III. "Avoiding Information Overload," *Military Review,* Headquarters, Department of the Army. Prepared by U.S. Army Command and General Staff College, Volume LXXVIII - July-August 1998, No. 4.

moving quicker through the OODA Loop. Both Boyd and Lind postulated that if U.S. ground commanders could see, think and then act faster than their adversaries, they could hit their adversaries before they were ready, or place them into a position which they were not prepared to accept. With thanks to Dr. Hayes and Dr. Signori, it appears appropriate to assume that the OODA Loop model could indeed be re-oriented and used as a tool in information assurance and knowledge superiority.

In its most basic form, one can see that today's fighter pilots and ground troops are not the only ones who can perform the functions of "observe, orient, decide, and act" to prosecute military operations. History shows us that even Alexander the Great was better at analyzing, deciding, and controlling his engagements—and he prevailed in nearly every conflict. To master the OODA Loop in today's environment, decision-makers will need the help of technology to obtain more or better information. Technology has the ability to mature the concept of the OODA Loop far beyond what Boyd had ever envisioned. But this technology now forces us to solve two fundamental challenges if we expect to dominate knowledge superiority within the battlespace. First, the proliferation of un-integrated, military, warfighting architectures gives the decision-makers potentially conflicting perspectives of the battlespace and thus introduces an exploitable vulnerability. Second, the explosion of available data creates an environment within the cognitive hierarchy that leads to information overload and hence to flawed decision making.

Regarding the first challenge, the large number of specialized, and often non-interoperable, warfighting architectures makes the integration of information to support overall coordination and control more important and more difficult. The second challenge is to harness that combat information explosion, thus improving decision-making. Recent exercises reveal an alarming number of unread messages, email and other such communications because of information overload. As the quantity of data rises, the difficulty of preparing, disseminating, digesting, interpreting and acting upon it grows. Traditionally, the military attempted to solve this problem by increasing the number of communications nodes. These past solutions only injected additional inputs and information without improving decision-making capability. The optimum solution must integrate the functions within the OODA Loop and give the decision-maker the correct dataset filtered through the cognitive hierarchy.

3.   COGNITIVE HIERARCHY. To gain advantage over the adversary's own OODA Loop, the decision-maker is faced with the problem of shortening the life-cycle of the decision-making process without increasing the failure rate of the decisions being made. That is, the decision-maker needs to secure an objective knowledge of the battlespace before his adversary does. This "perceptual" input will come from many sources and will begin to form a picture in the mind of the decision-maker. The object-oriented picture that is forming (data) will then be used to obtain information (forces, systems, tactics, etc.), and analysis of that information will then be used to gain knowledge (e.g., force placement), understanding (adversary intent), and awareness (what direction will the engagement take next).

One can sense that the OODA Loop cycle would be slowed by a growing deluge of data that are insignificant or not applicable to the task at hand. The difficulty lies in being able to percolate up through the cognitive hierarchy the exact bits and bytes of data that are useful. This filtering process can be pictured as a pyramid with the wealth of "data" laying the broad foundation for what will eventually reach the top—the wisdom that comes from having filtered the right data. Unfortunately, most military decision-makers possess limited time (driven by the OODA Loop) to perform specific tasks and issue orders. This is especially evident during warfighting exercises and operations. Further, as increased volumes of data are input into the base of the pyramid or as the rate of input increases, natural defense mechanisms try to protect the decision-maker from information overload.[4] A key method is a "bounded

---

[4] Jeffrey McKitrick et al., *The Revolution in Military Affairs*, Air War College Studies in National Security: Battlefield of the Future, no. 3 (Maxwell AFB, Ala.: Air University Press, September 1995), 65-97.

rationality"[5] that allows decision-makers to screen out inputs prior to being overloaded or inundated so that they can continue to focus on a particular task. One danger lies in the decision-maker screening out "golden nuggets" because they are focused elsewhere. A second danger lies in failing to recognize when new data should dictate a refocus or reorientation. As we mentioned earlier, recent operational exercises revealed an alarming number of unread messages, email and other such communications that might have guided that recognition. A quick review of the authors perception of the "cognitive," or in the case of military operations, "Command," hierarchy follows (Figure 1).
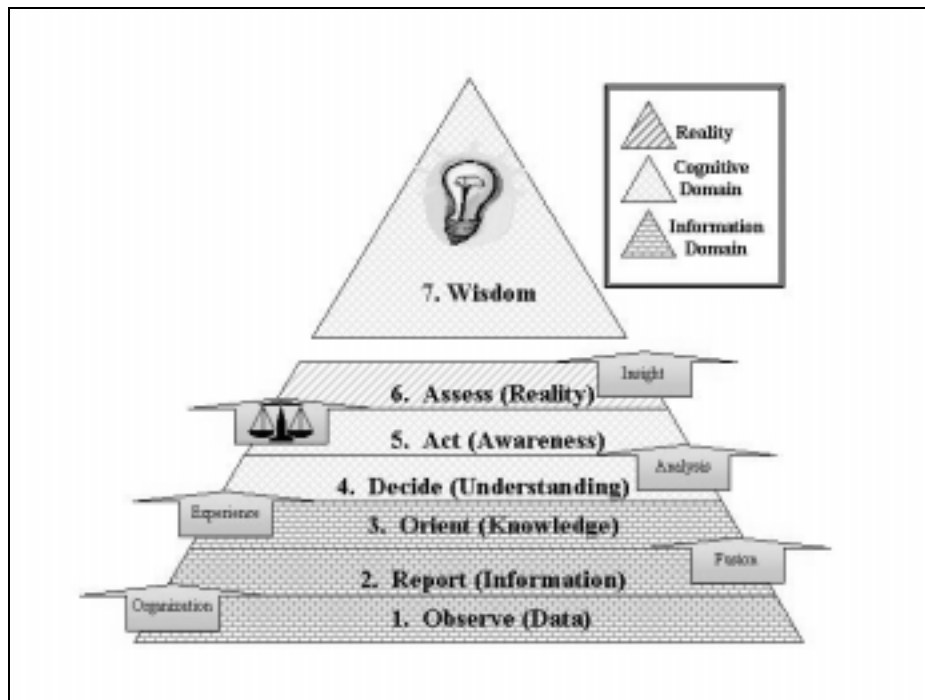

Figure 1. Command Hierarchy.

- Level 1: Data – Raw data is collected, and thus observed, from one or more sources. These data can eventually be augmented by rules imbedded in an expert system, or through population of large, separately maintained data structures. To reach the next level the data would have to be organized into information. In other words, data correlated becomes information.
- Level 2: Information – Data organized into some form that is useful to a human operator. Must be reported in a meaningful, recognizable form. To attain the next level, one must be able to fuse/integrate multiple information sources to form knowledge. In other words, fused information from multiple sources becomes knowledge.
- Level 3: Knowledge – Information integrated from multiple sources. To attain the next level, one must add common, environmental, real world experience to arrive at understanding. In other words, specific knowledge orients the decision-maker in real world settings and is used to predict the consequences of actions. This leads to understanding.
- Level 4: Understanding – Knowledge as personalized for the decision-maker and the situation, allowing the formulation of sound decisions. To attain the next level, analysis, cognitive agility, and feedback from others yields awareness.

---

[5] Herbert A. Simon, *Administrative Behavior: A Study of Decision-Making Processes in Administrative Organization* (New York: The Free Press, 1976), 38-41.

- Level 5: Awareness – The decision-maker's perception of reality based on forming a picture in his/her mind (sometimes referred to as "Having The Big Picture"). This "Big Picture" is a balance between one's own personal "view of the world" and the perceptions and inputs of those having close contact with the decision-maker (e.g., analysts and tacticians). This is where the decision-maker takes action.
- Level 6: Reality – This is the "real world." The closer that the decision-maker's "Big Picture" matches when overlaid onto the picture of the real world, the better the decision-making. Insight progresses from reality to wisdom. Reality, of course, includes the world of those not in close contact with the decision-maker (e.g., strategists, politicians) and information about which he or she may not be aware. At this point we are back in observation mode to determine the results of our actions and to see how well our awareness matches the reality. "Lessons Learned" are usually derived at this point in the loop.
- Level 7: Wisdom – This encompasses a deeper understanding of real world constructs coupled with intellect, instinct and intuition. The decision-making events at this level become textbook cases in how to master the shortening of the OODA Loop to overcome any adversary.

Technology can integrate functions within the OODA Loop and speed up the cycle. It does this by creating decision support tools to alleviate the doubtful situation that exists when crucial nuggets of information are omitted from the individual's OODA Loop. The tools aid in managing information to fit how decision-makers actually form a picture in their minds, assess situations, and then issue orders.[6] The decision support tools can deal with inputs from a large variety of different, sometimes contradictory, ambiguous or incremental sources.

We have discussed the decision-maker forming a mental picture. It is much easier for humans to do that than to manipulate and interpret text in one's mind. Simply, a picture is worth a thousand words, and a decision-maker understands the significance of an icon much faster than having to read a couple of pages of text. Over the years, most of our early, text-based automated systems (e.g., CPM, DOS, Unix) have been converted to or augmented by a Graphics User Interface (GUI) (e.g., Windows, X-Windows, Apple OS) for just that reason. The authors believe that this and the emergence of the object-oriented paradigm are empirical indicators that humans find it easier to describe "views of the world" through the notion of objects rather than text.

4.   THE OBJECT-ORIENTED PARADIGM. Object-oriented (OO) concepts have evolved from three distinct disciplines: artificial intelligence, conventional programming languages, and database technology. The object-oriented paradigm first surfaced in the late 1960's with the introduction of the SIMULA programming language. The Smalltalk programming language was, until recently, the best known and most popular example of object-oriented languages. In recent years C++, JAVA and other OO or OO-like languages have become widespread.

The following events have probably had the most influence over object-oriented development:

- Advances in computer architecture, including more capable systems and hardware support for operating systems;
- Advances in programming languages; and,
- Advances in programming methods.

---

[6] Lt Col Michael L. McGinnis and Maj George F. Stone III, "Decision Support Technology," *Military Review* 74, no. 11 (November 1994): 68.

An object-oriented data model would be implemented in Level 1 (the foundation of our cognitive hierarchy) where we would manipulate objects instead of records. The basic goal of this object-oriented database would be to model a real-world entity or object with a corresponding object in the database. Object-oriented databases are better at storing complex information and real-world data than are previous constructs. Much has been written on the Object-Oriented paradigm. The interested reader can find complete and detailed discussions in several of the references sited at the end of this paper. David A. Taylor[7] provides a very readable synopsis while Peter Coad[8] and Derek Coleman[9], among others, take a more in-depth look. What follows here is a very brief, high level overview.

4.1   The Concept. An OO system has a single type of entity, the *object*. An object is an entity whose behavior is characterized by the actions performed on and/or by it. "An object is an abstract mechanism that defines a protocol through which users of the object may interact."[10] Objects represent entities and concepts from the application domain being modeled and can be thought of as an identifier or type. Simply put, an object is an abstraction of a set of real-world things such that: all of the real-world things in a set (the instances) have the same characteristics, and all instances are subject to and conform to the same rules.

Most object-oriented systems make a distinction between the description of an object and the object itself. In a pure OO language or system, all conceptual entities are modeled as objects, described by attributes and behave as encapsulated in *methods*.

An *attribute* is an abstraction of a single characteristic possessed by all the entities that were, themselves, abstracted as objects. Simply, attributes describe objects. Attributes generally fall into three categories: descriptive, naming, and referential.

- *Descriptive attributes* are the intrinsic characteristics of an object. They provide facts indigenous to each instance of the object. Descriptive attributes of a warship, for example, may be length, type, mission, armament, crew complement, etc.
- *Naming attributes* provide facts about the arbitrary labels and names carried by each instance of an object, such as name, or identification number.
- *Referential attributes* capture the facts that tie an instance of one object to an instance of another object. In the warship case, the type of warship might be used as a referential attribute between class of warship and the type of engine installed on that class of ship.

An *identifier* is a set of one or more attributes that uniquely distinguishes each instance of an object. An object may have several identifiers any one of which can be used as a key.

A *method* is a function, capability, algorithm, formula, or process that an object is capable of performing. A method has five properties[11]:

- the generic function that it specializes;

---

[7] Taylor, David A. Object Technology: A Manager's Guide, 2nd Ed. Reading, MA: Addison-Wesley, 1997.
[8] Coad, Peter, and Edward Yourdon. Object-Oriented Analysis. Englewood Cliffs, NJ: Yourdon Press, 1990.
[9] Coleman, Derek, et. al. Object-Oriented Development: The Fusion Method. Englewood Cliffs, NJ: Prentice Hall, 1994.
[10] 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.
[11] Kim, Won and Frederick H. Lochovsky, eds. Object-Oriented Concepts, Databases, and Applications. Reading, MA: Addison-Wesley, 1989.

- its applicability condition;
- any qualifiers that identify the method's role;
- a parameter list that receives the arguments; and,
- the body executed when the method is called.

All of the action in object-oriented systems comes from sending *messages* between objects. Message sending is a form of indirect procedure call. A message must be sent to an object in order to find out anything about it.

Object-Oriented Design (*OOD*) is the process of decomposing a program/system into objects and establishing the relations between them. It is neither a function nor a data decomposition process. Instead, it seeks to identify those objects in the real world that must be manipulated to effect a decision by the user.

An *Object-Oriented Database Management System* (*OODBMS*) is a system that provides database-like support for objects (i.e., encapsulation and operations). It is a persistent, shareable repository, and manager of an object-oriented database. The database itself is a collection of objects defined by an object-oriented data model (objects that capture the semantics of objects supported in object-oriented programming)[12]. While semantic models are oriented toward structural abstraction and data representation, object-oriented models are concerned with behavioral abstraction and data manipulation.

An OODBMS attempts to extend flexibility to "unconventional" data and associated processing tasks (including text, graphics, and voice data) that cannot be handled and integrated by conventional database systems.

The basic idea of an object-oriented database is to represent an item in the real world with a corresponding item in the database. Coupling an object-oriented database with an object-oriented programming style results in the virtual elimination of the semantic gap between a program and the supporting data. Three levels of "object orientation" have been defined by Dittrich[13] and Manola[14]:

- ***Structurally object-oriented*** - the data model allows definitions of data structures to represent entities of any complexity (*complex objects*).
- ***Operationally object-oriented*** - the data model includes generic operators to deal with complex objects in their entirety.
- ***Behaviorally object-oriented*** - the data model incorporates features to define arbitrarily complex object types together with a set of specific operators (*abstract data types*). Instances can only be used to call these operators, their internal structure may only be exploited by the operator implementations.

Key features of systems that truly support the object-oriented philosophy as described by Cox[15,16]:

---

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

[13] 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.

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

[15] 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.

- *Inheritance* - instance variables, class variables and methods are passes down from a superclass to its subclasses. A technique that allows new classes to be built on top of older, less specialized classes instead of being rewritten from scratch.
- *Information Hiding* - the state of a software module is contained in *private variables*, visible only from within the scope of the module. Important for ensuring reliability and modifiability of software systems by reducing inter-dependencies between components.
- *Dynamic Binding* - the responsibility for executing an action on an object resides within the object itself. The same message can elicit a different response depending upon the receiver.
- *Encapsulation* - a technique for minimizing interdependencies among separately written modules by defining strict external interfaces. The user no longer applies operators to operands while taking care that the two are type compatible.
- *Data Abstraction* - the behavior of an abstract data object is fully defined by a set of abstract operations defined on the object. Objects in most object-oriented languages are abstract data objects. Can be considered a way of using information hiding.
- *Object Identity* - each object has a unique identifier independent of the values of properties.

Related notions currently associated with the object-oriented approach include[17] messages, overloading, late binding, and interactive interfaces with windows, menus and mice.

4.2   Advantages. Object-oriented programming, and database management systems offer a number of important advantages over traditional control/data oriented techniques, including, as described by Manola[18], King[19], and Thomas[20]:

- 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.
- The inheritance mechanism of object-oriented languages which allows code to be reused in a convenient manner.
- Facilitates the construction of software components that loosely parallel the application domain.
- Encourages the use of modular design.
- Provides a simple and expressive model for the relationship of various parts of the system's definition and assists in making components reusable or extensible.
- Views a database as a collection of abstract objects, rather than a set of flat (though possibly interrelated) tables.
- Captures integrity constraints more easily.
- Offers a unifying paradigm in the database, programming language, and artificial intelligence domains.

---

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

[17] 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.

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

[19] 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.

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

- The ability to represent and reference objects of complex structures resulting in increased semantic content of databases.
- Provides a more flexible modeling tool.
- Allows protection and security mechanisms to be based on the notion of an object, a more natural unit of access control.
- Can provide version control functions.
- Incorporation of software engineering principles such as data abstraction and information hiding.

4.3   Disadvantages. Despite its many advantages, the object-oriented view is not perfect. However, though there are several drawbacks to OO systems listed below, most are a direct result of its relative infancy and lack of development. Most of these problems are expected to be resolved as the model matures, as written by Andrews[21], Kim[22] and Manola[23].

- The object-oriented paradigm lacks a coherent data model. There is currently no established standard for object-oriented design, methodology, language facilities, etc.
- Research into structures for efficient object storage is in the early stages of development.
- Use of an object-oriented database from a conventional data processing language is difficult because of the semantic gap.
- In current environments, the run-time cost of using object-oriented languages is high.
- Object-oriented database management systems provide only limited support for integrating data in existing, possibly heterogeneous, databases.
- Typical OODBMS's do not integrate existing database application code with the object methods maintained by the system. This is similar to the previous point but concerns procedures rather than data.
- Many object-oriented database systems do not support the strict notion of metaclass.
- Some OODBMS's do not emphasize the efficient processing of set-oriented queries, although most of the commercial OODBMS's provide some form of query facility.
- Object-Oriented Programming / Processing (OOP) can be very memory intensive.
- There is no adequate, accepted, standard query language based on the OO view of data.

Many of these issues are well on their way to resolution. For example, the Unified Modeling Language (UML) seems to be emerging as a standard[24], filling the gap mentioned in the first bullet above. The OO concept has already made great strides. As the paradigm matures, most, if not all, of these issues are expected to be resolved.

4.4   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

---

[21] 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.

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

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

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

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 OODA Loop, 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.

4.5   Building Blocks. There is a need for a common description for architectures. Architectures are typically developed from the functional requirements (in the case of notional architectures) or functional capabilities (in the case of physical, existing architectures) that we consider essential to warfighting. These requirements have been expressed in many forms and at varying levels of granularity. However, if we adopt the concept of *functional requirements*[25] 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 2).
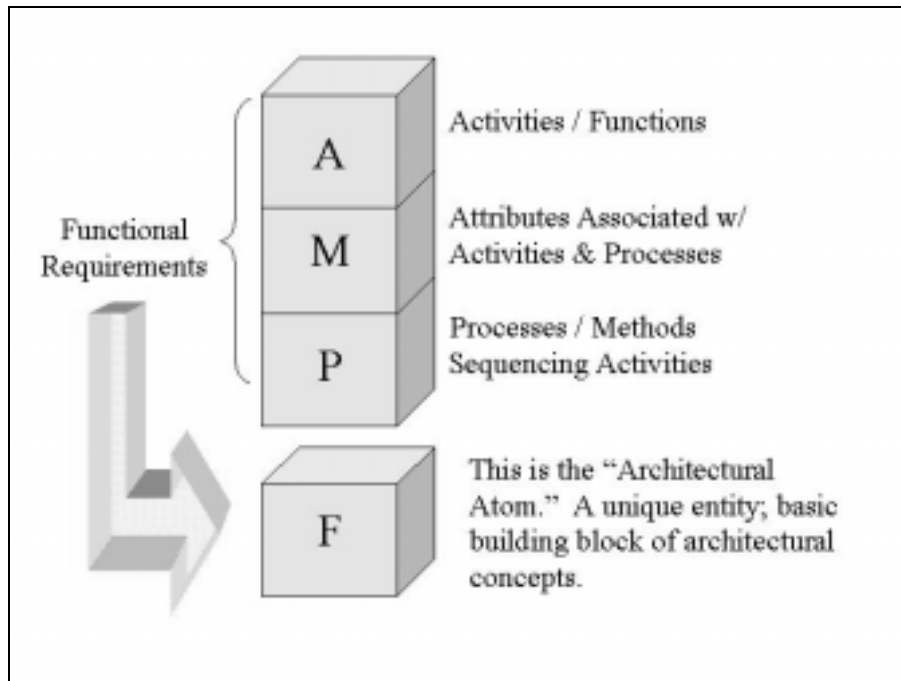

Figure 2. Basic Building Blocks.

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, there are attributes associated with that function, and processes, methods or activities are performed by that function.

In this definition, Functional Requirement Objects become "Architectural Atoms," the building blocks from which any and all architecture components can be constructed.

---

[25] The term "functional requirements" will be used henceforth to include both the functions that we wish to perform (requirements) and the functions that existing systems actually do perform (capabilities).

These components become systems and systems, in turn form a system of systems, or suite (Figure 3). 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 systems. Some redundancies, however, may be targeted for elimination in the name of efficiency and/or cost effectiveness.
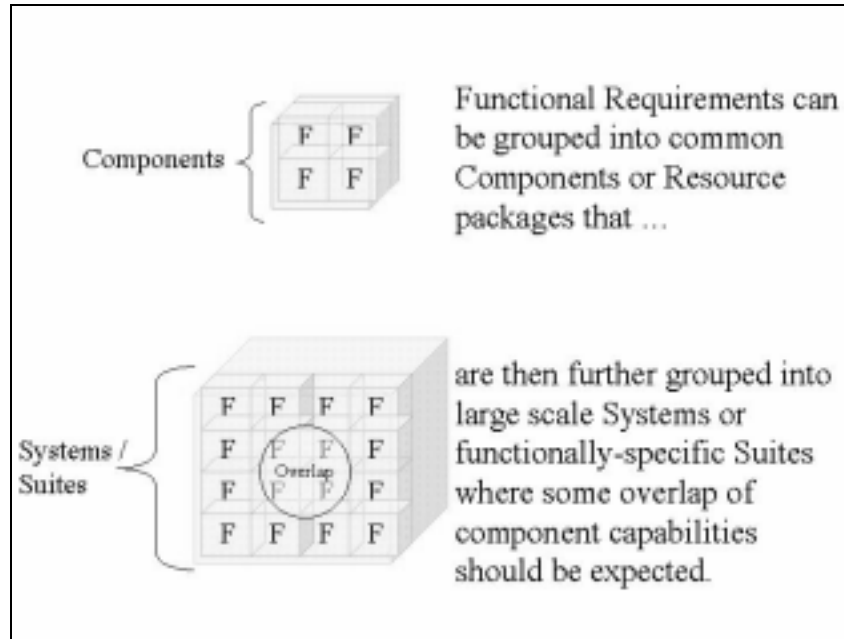

Figure 3. Functions and Components.

Thus, from a functional perspective, the entire architecture can be described using combinations of Functional Requirements (Figure 4).
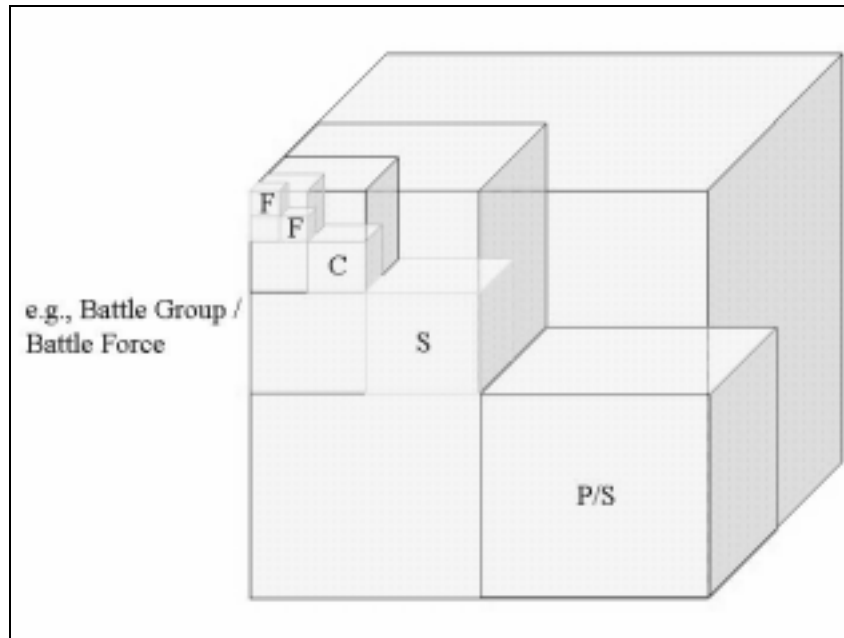

Figure 4. Building Blocks.

Object-Oriented architectural components, when assembled, might resemble a Rubik's Cube (Figure 5). 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.
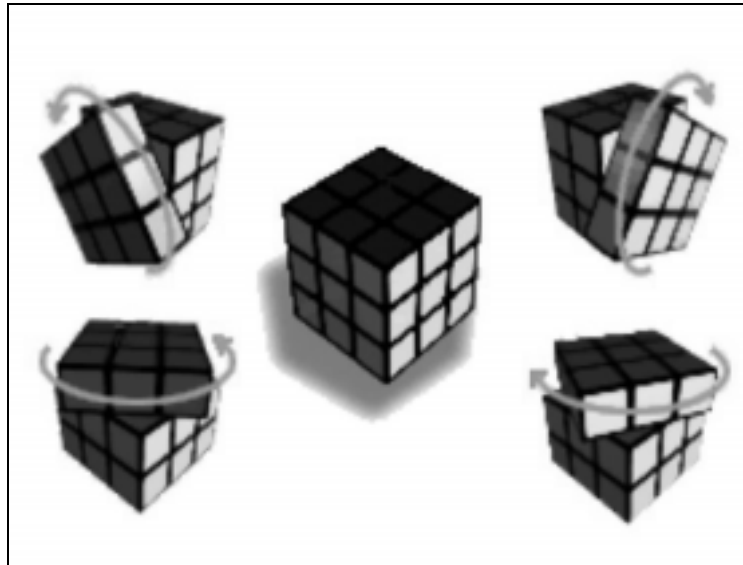

Figure 5. Rubik's Architecture Cube.

This is one of the major disappointments with architectures today and a primary reason that our systems are still not interoperable despite more than ten 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.[26, 27]

Traditional database systems are limited in their data abstraction and representation power, and they fall short of providing important information management capabilities required by certain applications such as office information systems, personal databases, design engineering databases, and artificial intelligence systems. The use of object-oriented methodologies to support the decision-maker at various levels of abstraction is an important emergent area where great strides can be made.

---

[26] 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.

[27] 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.

5.0   SUMMARY. So, how does this concept of Object-Oriented Architecture help us achieve a higher level within the cognitive hierarchy and what will that do for the speed and precision with which we navigate the OODA Loop?

All things start with raw data elements, just like 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 climb the cognitive 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 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, centralizing and utilizing one common dataset solves many of the problems that we have discussed here and many more that were only alluded to.   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 to ensure that everyone is working with the same "big picture," all singing from the same sheet of music.
3. Ensure that the architectural data in centralized, standardized, common and available to all that need it. This makes for great strides toward solving the interoperability issue. Ensure someone is paying close attention to the interfaces so that they can at least compare apples to apples.
4. Allow for a more efficient 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. Collect, organize and mature these data so that we can begin to climb the cognitive hierarchy, learning more and more about ourselves, our systems and our capabilities. If we apply the same rigor to threat data, we can easily produce a better understanding of our adversaries, their systems and our relative standing in any conflict or engagement.
6. This higher understanding leads to a heightened level of awareness that allows us to navigate that all important OODA loop quicker and more efficiently than our adversaries.

Thus, by attacking and resolving the lowest level problem (the Architectural Atoms), 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.
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 task mentioned above probably present the biggest challenges. Although we are getting much better at Joint organizations and operations, we continue to have political 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 to design, develop and maintain its own separate architectural data and data structures, even if we can successfully prescribe a standard format. 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 system will fulfill the same goal if properly implemented.

The biggest and most costly challenge will likely remain the collection of the huge amounts of data required to drive such a system. However, if we consider the large amounts of money that are spent today by a very large 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 experience today. Costs notwithstanding, what we have shown in this paper is that it is quite possible for a decision-maker to be able to compare or otherwise manipulate large amounts of data in order to "observe, orient, decide and act" without achieving information overload through an object-oriented "view of the world."

Obituary - Col. John Boyd

9 MAR 1997. WEST PALM BEACH, Fla. - Col. John R. Boyd, an Air Force fighter pilot whose belief that quicker is better than faster became the basis of a theory that revolutionized military strategy, died of cancer today. He was 70.

Boyd flew only a few combat missions in Korea. But after wondering why the comparatively slow U.S. F-86's almost totally dominated the superior MiG-15 fighter, he figured out the F-86's advantages: better visibility and a faster roll rate.

Boyd theorized that the key to victory was not a plane that could climb faster or higher, but one that could begin climbing or change course quicker.

From 1954 to 1960, Boyd, who helped establish the Fighter Weapons School at Nellis Air Force Base in Nevada, had a standing offer to pilots: take a position on his tail, and after 40 seconds of twists and turns he would have the challenger in his sights or pay $40. He never lost the bet.

He was assigned to the Pentagon in 1964. Boyd's design ideas helped give the F-15 a big, high-visibility canopy. But his major triumph was the F-16, which is far more agile and costs half as much.

Though his writings on the subject of decision cycles and OODA Loops were seminal and prolific, the majority of his works were never published.

Bibliography

1. Abriel, J. R. <u>Data Semantics</u>. Amsterdam, Netherlands: North-Holland, 1974.

2. Association for Computing Machinery. <u>Proceedings: Conference on Object-Oriented Programming, Systems, Languages, and Applications</u>. New York, NY: ACM, 1986.

3. Alagic, Suad. <u>Object-Oriented Database Programming</u>. New York, NY: Springer-Verlag, 1988.

4. Alashqur, A. M., S. Y. W. Su, and H. Lam. "OQL: A Query Language for Manipulating Object-Oriented Databases." <u>Proceedings of the Fifteenth International Conference on Very Large Databases</u>. Amsterdam, Netherlands: 1989.

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

6. Bailin, Sidney C. "An Object-Oriented Requirements Specification Method." <u>Communications of the ACM</u>, 32:5 (May), 608-623, 1989.

7. Banerjee, Chou, Garza, Kim, Woelk and Ballou. "Data Model Issues for Object-Oriented Applications." <u>ACM Transactions on Office Information Systems</u>, 5:1 (Jan), 3-26, 1987.

8. Bateman, Robert L. III. "Avoiding Information Overload," *Military Review,* Headquarters, Department of the Army. Prepared by U.S. Army Command and General Staff College, Volume LXXVIII - July-August 1998, No. 4.

9. Booch, Grady. "Object-Oriented Development." <u>Tutorial: Object-Oriented Computing, Volume II: Implementations</u>, 5-15. Gerald E. Peterson, ed. Washington, DC: Computer Society Press, 1987.

10. Boyd, J. <u>Patterns Of Conflict</u>, December 1986. Unpublished study, 196 pages.

11. Coad, Peter, and Edward Yourdon. <u>Object-Oriented Analysis</u>. Englewood Cliffs, NJ: Yourdon Press, 1990.

12. Codd, E. F. "A Relational Model of Data for Large Shared Data Banks." <u>Communications of the ACM</u>, 13:6 (Jun), 377-387, 1970.

13. Codd, E. F. "Extending the Database Relational Model to Capture More Meaning." <u>Transactions on Database Systems</u>, 4:4 (Dec), 397-434, 1979.

14. Cohen, Benjamin. "Merging Expert Systems and Databases." <u>AI Expert</u>, 4:2 (Feb), 22-31, 1989.

15. Coleman, Derek, et. al. <u>Object-Oriented Development: The Fusion Method</u>. Englewood Cliffs, NJ: Prentice Hall, 1994.

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

17. Cox, Brad J. <u>Object Oriented Programming</u>. Reading, MA: Addison-Wesley, 1986.

18. Curts, Raymond J. "A Systems Engineering Approach to Battle Force Architecture." Unpublished research. Fairfax, VA: 1989.

19. Curts, Raymond J. "An Expert System for the Assessment of Naval Force Architecture." Unpublished research. Fairfax, VA: 1989.

20. Curts, Raymond J., and Campbell, Douglas E. "Architecture: The Road to Interoperability." <u>Proceedings: 1999 Command & Control Research & Technology Symposium (CCRTS)</u>, U.S. Naval War College, Newport, RI, June 29-July 1, 1999.

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

22. Derrett, N., W. Kent and P. Lunghaek. "Some Aspects of Operations in an Object-Oriented Database." <u>IEEE Database Engineering Bulletin</u>, 8:4 (Dec), 1985.

23. Dittrich, Klaus R. "Object-Oriented Database Systems: The Notion and the Issue." <u>International Workshop on Object-Oriented Database Systems</u>. Washington, DC: Computer Society Press, 1986.

24. Defense Systems Management College. <u>DSMC Systems Engineering Management Guide</u>. Washington, DC: Government Printing Office, 1986.

25. Egan, John T. "EW Architecture Assessment Methodology." Washington, DC: Unpublished Research, 1989.

26. Fowler, Martin with Kendall Scott. <u>UML Distilled: Applying the Standard Object Modeling Language</u>. Reading, MA: Addison-Wesley, 1997.

27. Gaines, B. R. "Methodology in the Large: Modeling All There Is." <u>Proceedings: International Conference on Cybernetics and Society</u>. New York, NY: IEEE Press, 1956.

28. <u>Handbook for Decision Analysis</u>. Washington, DC: Defense Advanced Research Projects Agency, 1977.

29. Hodges, Parker. "A Relational Successor?" <u>Datamation</u>, 35:21 (Nov), 47-50, 1989.

30. Hong, S. and F. Maryanski. "Representation of Object-Oriented Data Models." <u>Information Science</u>, 1988

31. Hull, Richard and Roger King. "Semantic Database Modeling: Survey, Applications, and Research." <u>ACM Computing Surveys</u>, 19:3, 201-260, 1987.

32. <u>Jane's All the World's Weapon Systems</u>. London, GB: Jane's, 1989.

33. Kerschberg, Larry and Joao E. S. Pacheco. "A Functional Data Base Model". Rio de Janeiro, Brazil: Pantificia Univ. Catalica, 1976.

34. Kerschberg, Larry, ed.. Expert Database Systems: <u>Proceedings of the 1st International Workshop</u>. Menlo Park, CA: Benjamin Cummings, 1987.

35. Kerschberg, Larry, ed. <u>Expert Database Systems: Proceedings of 2nd International Conference</u>. Redwood City, CA: Benjamin Cummings, 1988.

36. Kerstin, Martin L. and Frans H. Schippers. "Towards an Object-Centered Database Language." <u>Proceedings: International Workshop on Object-Oriented Database Systems</u>. Klaus Dittrich and Umeshwar Dayal, eds. Washington, DC: Computer Society Press, 1986.

37. Khoshafian, S. N. and G. P. Copeland. "Object Identity." New York, NY: ACM, 1986.

38. Kim, Won and Frederick H. Lochovsky, eds. <u>Object-Oriented Concepts, Databases, and Applications</u>. Reading, MA: Addison-Wesley, 1989.

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

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

41. Lam, Xia, Qiu, Wu, Chen, Pant, Geum and Su. "Prototype Implementation of an Object-Oriented Knowledge Base Management System." PROCIEM, 1989.

42. Lam, H., S. Y. W. Su and A. M. Alashqur. "Integrating the Concepts and Techniques of Semantic Modeling and the Object-Oriented Paradigm." COMPSAC, 1989.

43. Maier, David, J. Stein, A. Otis and A. Purdy. "Development of an Object-Oriented Database Management System." New York, NY: ACM, 1986.

44. Manola, F. and U. Dayal. "PDM: An Object-Oriented Data Model." New York, NY: IEEE, 1986.

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

46. McGinnis, Lt Col Michael L. and Maj George F. Stone III, "Decision Support Technology," *Military Review* 74, no. 11 (November 1994): 68.

47. McKitrick, Jeffrey et al., "The Revolution in Military Affairs," <u>Air War College Studies in National Security: Battlefield of the Future</u>, no. 3 (Maxwell AFB, Ala.: Air University Press, September 1995), 65-97.

48. Mellor, Stephen J., and Sally Shlaer. <u>Object-Oriented System Analysis: Modeling the World In Data</u>. Englewood Cliffs, NJ: Yourdon Press, 1988.

49. Meyer, Bertrand. "Reusability: The Case for Object-Oriented Design." <u>IEEE Software</u>, 4:2 (Mar), 50-64, 1987.

50. Orr, George E. Combat Operations C3I: Fundamentals and Interactions. Maxwell AFB, AL: Air University Press, 1983.

51. Parsaye, Chignell, Khoshafian, and Wong. Intelligent Databases: Object Oriented, Deductive Hypermedia Techniques. New York, NY: Wiley & Sons, 1989.

52. Peckham, Joan and Fred Maryanski. "Semantic Data Models." ACM Computing Surveys, 20:3 (Sep), 153-189, 1988.

53. Planning Research Corporation. Naval Warfare Tactical Data Base User's Guide. McLean, VA: PRC, 1988.

54. Shlaer, Sally and Stephen J. Mellor. Object-Oriented Systems Analysis: Modeling the World in Data. Englewood Cliffs, NJ: Yourdon Press, 1988.

55. Sibley, E. H. and L. Kerschberg. "Data Architecture and Data Model Considerations." Reston, VA: AFIPS Press, 1977.

56. Simon, Herbert A., Administrative Behavior: A Study of Decision-Making Processes in Administrative Organization. New York: The Free Press, 1976, 38-41.

57. Smith, J. M. and D. C. P. Smith. "Database Abstractions: Aggregation and Generalization." ACM Transactions on Database Systems, 2:2 (Mar), 105-133, 1977.

58. Sowa, John F. Conceptual Structures. Reading, MA: Addison-Wesley, 1983.

59. Space & Naval Warfare Systems Command (SPAWAR). "Weapon System Architecture and Engineering: History and Mission." Unpublished Research. Washington, DC: SPAWAR, 1987.

60. Space & Naval Warfare Systems Command (SPAWAR). "Space & Naval Warfare Systems Command Consolidated Glossary." Washington, DC: SPAWAR, 1987.

61. Space & Naval Warfare Systems Command (SPAWAR). Physical Architecture for Sea Control for the Conduct of Electronic Warfare, Volume I, Part A. Washington, DC: SPAWAR, 1989.

62. Space & Naval Warfare Systems Command (SPAWAR). Functional Architecture for Sea Control for the Conduct of Electronic Warfare, Volume I, Part B. Washington, DC: SPAWAR, 1989.

63. Space & Naval Warfare Systems Command (SPAWAR). Battle Group Data Base Management (BGDBM). Washington, DC: SPAWAR, 1989.

64. Stonebraker, Michael, and Lawrence A. Rowe. "The Postgres Papers." Berkley, CA: University of California, 1987.

65. Su, S. Y. W., V. Krishnamurthy and H. Lam. "An Object-Oriented Semantic Association Model (OSAM*)." Artificial Intelligence: Manufacturing Theory and Practice. Kumara, Soyster and Kashyap, eds. Norcross, GA: Industrial Engineering Press, 1989.

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

67. Thomas, Dave. "Object-Oriented Databases and Persistent Objects." Journal of Object-Oriented Programming, 2:2, 59-60, 1989.

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

69. Wiersma, R. "Warfare Systems Architecture and Engineering Process." Unpublished Research. Dahlgren, VA: Naval Surface Warfare Center, 1987.

70. 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.

Vita

CDR Raymond J. Curts, Ph.D., (USN, Ret.) was born December 2, 1946 in Philadelphia, Pennsylvania and is an American citizen. He graduated from Vandalia Community High School, Vandalia, Illinois in 1965. He received his Bachelor of Science in Aeronautical and Astronautical Engineering from the University of Illinois in 1970 and was commissioned as an Ensign in the United States Navy. In December 1972 he earned his wings as a Naval Aviator and was assigned to the U.S. Naval Base at Guantanamo Bay, Cuba. Returning to the continental United States in 1976, he became an instructor pilot in the Navy's Advanced Jet Training Command in Beeville, Texas where he earned a Master of Arts degree in Management and Business Administration from Webster College of St. Louis, Missouri. After tours of duty in Norfolk, Virginia; Rota, Spain; and Key West, Florida, he was stationed at the Space and Naval Warfare Systems Command (SPAWAR) in Washington, DC where he spent five years as the Navy's Electronic Warfare Architect. During this time he earned a Ph.D. in Information Technology from George Mason University. Since retirement from the Naval service in 1992, Dr. Curts has supported a wide variety of government agencies with several major corporations. Currently he is providing Information Assurance support to the Department of State, the U.S. Navy and several other government agencies.

LCDR Douglas E. Campbell, Ph.D., (USNR-R, Ret.) was born on May 9, 1954 in Portsmouth, Virginia, and is an American citizen. He graduated from Kenitra American High School, Kenitra, Morocco, in 1972. He received his Bachelor of Science degree in Journalism from the University of Kansas in 1976 and was immediately commissioned as an Ensign in the United States Navy. He joined the U.S. Naval Reserve Program as an Intelligence Officer in 1980 and was transferred to the Retired Reserves as a Lieutenant Commander on 1 June 1999. Dr. Campbell received his Master of Science degree from the University of Southern California in Computer Systems Management in 1986 and his Doctor of Philosophy degree in Computer Security from Southwest University in New Orleans, Louisiana, in 1990. Dr. Campbell is president and CEO of Syneca Research Group, Inc., a certified 8(a) and a certified Small & Disadvantaged Business entity under the U.S. Small Business Administration's program. He is also co-founder and Director of Syneca Foundation, a non-profit organization assisting American Indian students and their communities.