Defensible C4ISR Architectures Require the Application of Good System Engineering Practice

Steven H. Dam, Ph.D. and James D. Willis, Jr
Systems and Proposal Engineering Company
10060 Valley Dale Lane
Marshall, VA 20115
www.spec-1.com

ABSTRACT

The Department of Defense's (DoD) Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) Architecture Framework, version 2.0, was developed to facilitate comparison of like and dissimilar information systems architectures. These architectures could be complementary or competing systems. Having a common standard means for such comparisons is absolutely essential as the Department fosters more interoperability among their systems and makes tough priority decisions to make optimal use of its funding. To be compliant architects must produce operational, systems and technical views that contain particular sets of information. Although these could be produced in common word processing and presentation tools, the availability of system engineering tools and their application as part of good system engineering practice enhance accuracy and traceability of all information, as well as, ease change to that information as new data becomes available or the situation changes.

This paper describes how to apply good system engineering practice, in terms of methodology, tools, process and people, to develop defensible C4ISR Architectures and increase customer satisfaction.

BACKGROUND

In the early 1990's DoD had difficulty in comparing architectures developed by competing contractors who were trying to help the Department solve problems found during the Gulf War.[1] In particular, the systems associated with C4ISR were becoming increasingly complex and rapidly obsolete due to the explosion of capabilities provided by the information technology revolution. In addition, the services, commands, and DoD agencies had a penchant for designing and developing their own independent capabilities and solutions despite extraordinary similarities among the requirements, performance criteria, and applications contexts. At the end of these independent trails, the resulting systems would not work well together further complicating difficult joint operations.

To solve this problem, the Office of the Secretary of Defense (OSD) chartered the Architecture Working Group (AWG) to develop a means for comparing C4ISR architectures. This working group had representatives from every joint command, Service and Agency in DoD. Since many of these representatives had a background in system engineering, they applied system engineering terminology and products to create the architecture views. The result was Version 1.0 of the C4ISR Architecture

© 2002 Systems and Proposal Engineering Company. All Rights Reserved.

Framework, which was released in 1996. As with many initial versions, significant work was still needed, so the group pushed ahead with version 2.0 in late 1997. In early 1998 the Framework was mandated for use on all new architecture developments.[2] Work continues today and a new version is expected in October 2002 that will be titled "DoD Architecture Framework 1.0" to broaden its application to all DoD architectures. **Figure 1** summarizes this history.

The mandate, combined with common sense, suggests we need to produce these architecture views in a coherent, cost-effective manner. Since the Framework, contrary to popular opinion, *does not specify methodologies, tools, or processes*, the architect must choose and defend these factors. These factors employed by the appropriately skilled people are the foundation of a credible, defensible set of C4ISR Architecture Framework products. However, having accurate technical products does not guarantee that the decision makers will accept your results. Therefore, you must also craft the message in a form that the decision makers can quickly assimilate the results and make the correct decision (selecting and/or approving and funding your architecture, of course). The remainder of this paper describes these steps in transforming a simple compliance with the C4ISR standard into a complete system engineering analysis and resulting in greater acceptance by the decision maker and your customer.

In addition, a defensible architecture is more achievable and should more readily result in a system that satisfies the customer's requirements.

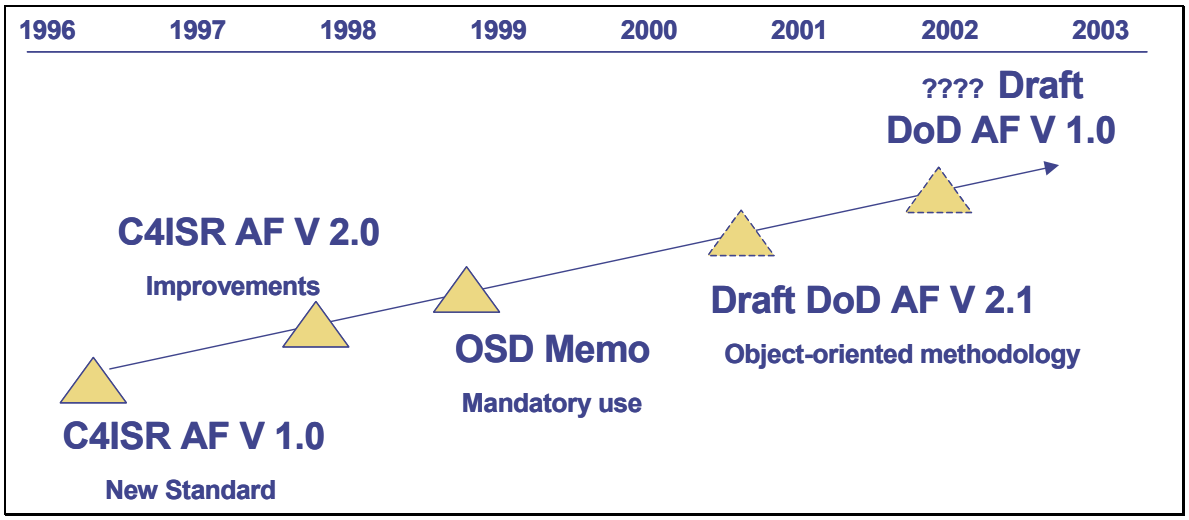

Figure 1. A Brief History of the C4ISR Architecture Framework

METHODOLOGY

The first step along the road to a defensible set of C4ISR products comes by defining your methodology. A methodology contains the rules for performing the necessary system engineering analyses upon which all other analyses and decisions are based. The text box on the next page shows some of the characteristics of what we consider a "good" methodology.

© 2002 Systems and Proposal Engineering Company. All Rights Reserved.

Characteristics of a Good Methodology

- Addresses your full life cycle
- Integrates a set of processes
- Provide executable results
- Uses appropriate software tools
- Communicates well to all audiences
- Extends ability to adjust to specific needs
- Has been applied successfully

Examples of methodologies abound in system engineering; and most of us have our own favorite. Some of the better-known methodologies share certain similarities, but are quite different in other respects. Some of the more well know include Structured Analysis[3] with and without real-time extensions, Object-Oriented Analysis and Design,[4] Unified Modeling Language (UML),[5] System/ Software Requirements Engineering Methodology/ Distributed Computer Design System (SREM/DCDS),[6] and the Zachman Framework.[7] Having used, or attempted to use, most of these methodologies, we have found all to have their pros and cons. We have selected the SREM/DCDS methodology as the "best in class" due to its well-defined schema, executable functional analysis and instantiation in multiple tools. The essence of the SREM/DCDS methodology can be seen in the form of the behavior diagram (see **Figure 2** for an example). The behavior diagram combines the data flow (e.g., IDEF0) and control (e.g., function flow block diagram) views of the function into a single diagram. Most methodologies treat these two views separately and thus miss critical synergistic effects. In addition, the SREM/DCDS methodology has been proven to be executable; hence it includes the "time dimension" in the analysis. This situation is analogous to the difference between compiling and executing software code. It's easy to design a system that looks logical, but doesn't execute. Many system design failures can be traced back to a lack of executability in the design.

The only "cons" about this methodology is the lack of a definitive book to reference and the fact that many people perceive these diagrams as too complex. The best access to this methodology today is in the course referenced in Reference 6. The perception problem is a general problem will be addressed in the last part of this paper.

Now some people might ask the question, "Why is functional analysis import to developing C4ISR Architecture Framework products?" If you look closely at many of the Framework products, they result from functional analysis. For example, the OV-3 (Operational View – Operational Information Exchange Matrix) would be difficult to derive for any system without having performed a functional analysis. Likewise, the OV-5 (Activity Model) and OV-6 (dynamic operational views) come from functional analysis. In the systems views, the SV-3 (System^2 Matrix), SV-4 (System Functionality Description), SV-5 (Operational Activity to System Function Traceability Matrix), SV-6 (System Data Exchange Matrix), and the SV-10 (dynamic systems views) all come from functional analysis. Although you can make the correspondence between these functional views and object-oriented or UML views, the writers of the Framework had basic functional analysis in mind when developing these views. Therefore, we chose to pursue the proven functional analysis approach in our architecture development projects.

© 2002 Systems and Proposal Engineering Company. All Rights Reserved.

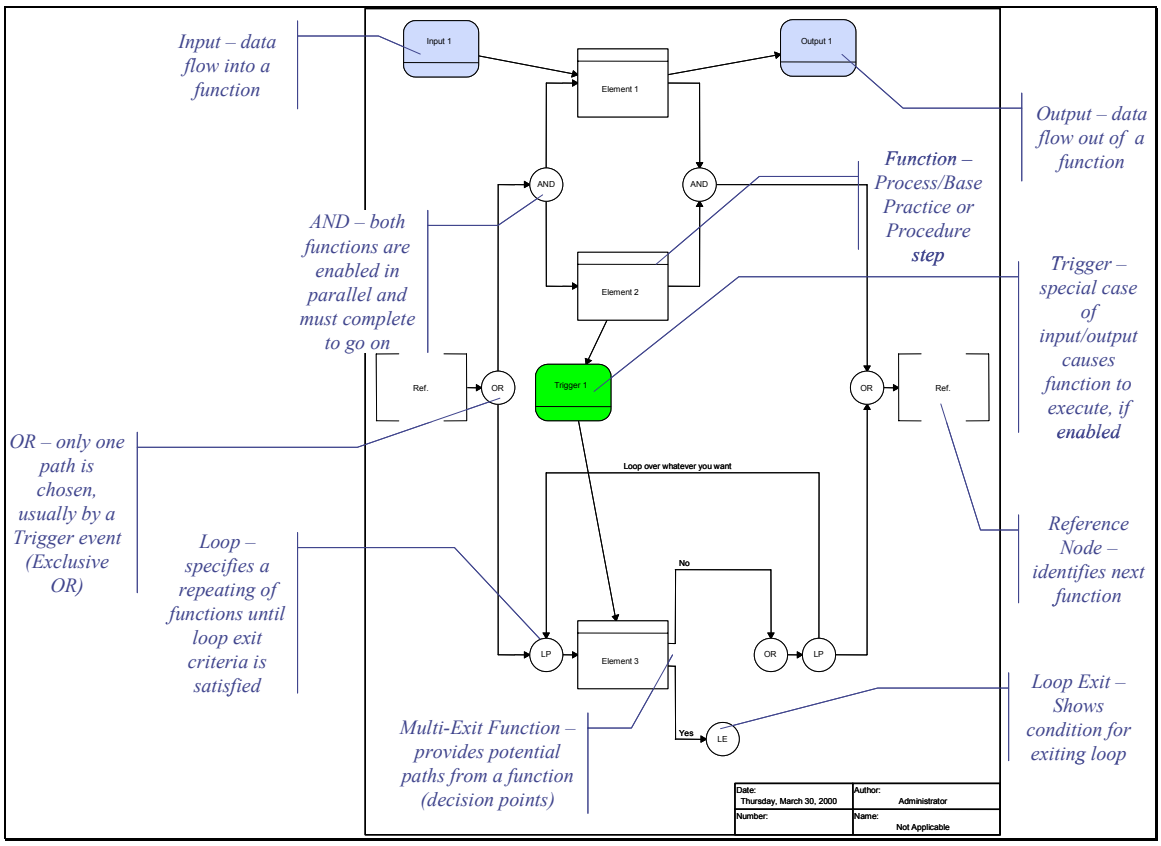

Figure 2. Behavior Diagrams Provide Executable Logic

TOOLS

Once you have selected a methodology, you need to select the software tools that will support that methodology. Some people know only Microsoft Office, and while Office may be a necessary part of your toolset, great value can be found in current system engineering tools. Most of the methodologies listed in the previous section use a large number of parameters, characteristics and/or attributes to describe the system fully. This raises the complex problem or system "How do you capture and control this information?" To keep track of these parameters, control changes, and create the drawing from them is very labor-intensive without the use of a database tool. Before the advent of personal computers, we had rooms of engineers whose sole job was to track all the paper, ensure its consistency, and make the never-ending changes to those documents. We now can employ a small system engineering team linked together through a common database tool to do the same job, thus reducing costs by an order of magnitude.

From our experimentation and experience we've identified several characteristics we believe are fundamental in a good tool. The text box on the next page lists some of the characteristics of a "good" tool. The more of these your tool has, the more robust is the tool.

For some years now we've been using Vitech Corporation's CORE® tool, which combines all the attributes above. CORE tightly couples requirements analysis, with functional analysis and simulation

© 2002 Systems and Proposal Engineering Company. All Rights Reserved.

Characteristics of a Good Tool

- Supports your chosen methodology
- Provides several integrated functions
- Employs rule-based standards
- Enforces consistency between views
- Uses an integrated, common database
- Permits simulation capability
- Facilitates exporting data and products
- Enables flexible reconfiguration
- Applies to multiple lifecycle phases
- Supports multiple disciplines

capabilities, operating across a common object-oriented database. The report generation capability also appeals to the programmer in that it provides a powerful scripting language and schema extension capability that the authors have employed successfully on a number of systems, software and architecture projects. This capability enhances the already robust schema and reports available with the tool.

Choosing a tool sometimes is driven by economics, as well as, practical functionality. In comparing prices of other tools, you must ensure you have identified all comparable functionality. This usually requires combining two or three tools, which may or may not be loosely coupled, to provide the same capability. In particular the built-in discrete event simulator can be started at anytime a behavior diagram (Vitech calls this an Enhanced Function Flow Block Diagram) has been created. The only comparable, single product was the RDD-100 tool by Ascent Logic Corporation, which has been discontinued.

Vitech has recently adapted SPEC's C4ISR Architecture schema for CORE and is now available in version 4.0. This schema enables the user to develop Framework-compatible documentation, while doing good system engineering.

Other tools can be used to enhance the presentation, such as netViz, Visio, or System Architect, but here's where the Microsoft Office suite provides most, if not all, of the capability needed to produce briefings and reports.

PROCESS

Having a methodology and tools are just the beginning. Choosing and applying an integrated set of processes for implementing the methodology and tools becomes critical to a successful architecture development project. Since most customers do not like having to pay for system engineering or studies (they need/want hardware and software now!), we need to produce the paper products as rapidly as possible, while maintaining the integrity and rigors of the analysis. A standard systematic process aids in reaching that goal by reducing the "what are we going to do next?" question that always arises in the heat of a short turnaround project (or proposal).

Figure 3 provides an example of a process in a "Gantt Chart" format. The steps are numbered to indicate their nominal start sequence. Notice that most of these steps overlap demonstrating the concurrent, iterative nature of this process. It goes slightly beyond the minimum requirements for developing C4ISR Architecture Framework products, since it assumes that this process is not an end in itself. It provides the necessary information for executing the resulting architecture, including the requirements analysis and operational demonstration. Step 15 recognizes that throughout any architecture project you will have to produce reports and briefings to show status and progress. By using the methodology and tools selected above, the authors have found that the development of reports and briefings is now a simpler – by-product of the process, instead of a driver.

© 2002 Systems and Proposal Engineering Company. All Rights Reserved.

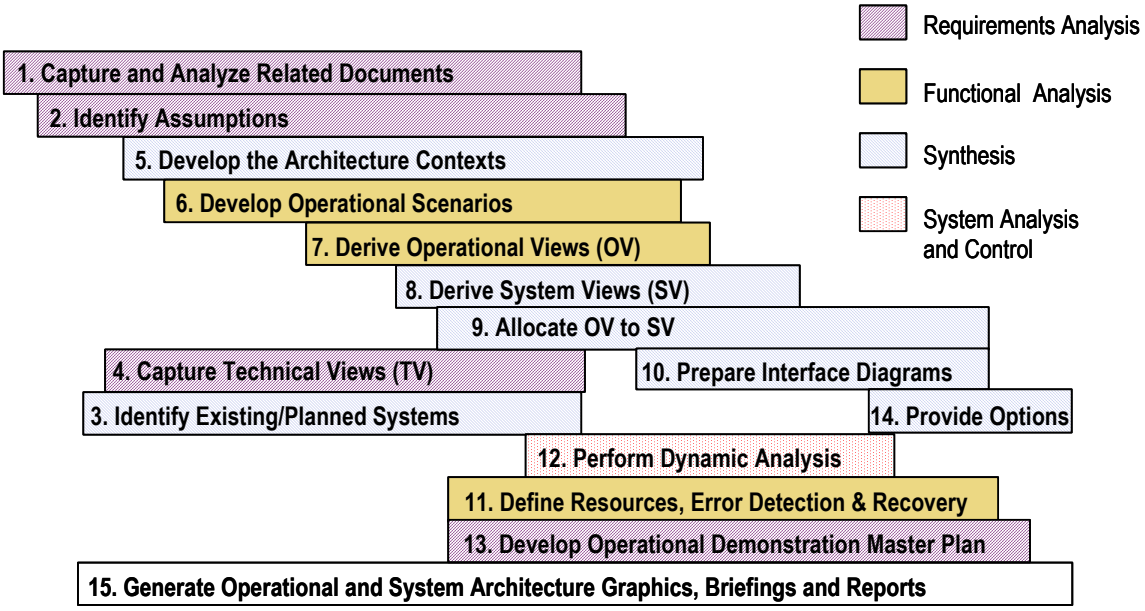

Figure 3. Develop a Process for Creating Architectures

PEOPLE

Perhaps the most important part of any architecture development is selecting the right people to perform the necessary analyses and express the results in a form understandable to all your audiences, which include engineers, evaluators, customers and contractors. The text box below shows some of the kinds of people you will need to execute a successful architecture project.

Who Should Be Part of My Architecture Team?

- Need someone with vision
- Need someone who can perform the detailed system engineering
- Need someone who understands the domain
- Need someone familiar with the methodology and tools
- Need someone who understands the process

Since no one person has all these attributes, team composition will become a critical factor in determining success or failure of the project. Very often it is difficult to find personnel with the experience and knowledge of specific methods, tools and processes. Too often engineers only receive "on-the-job training" in these items. A successful project requires that the personnel be trained before starting, not during or after. Hence, if any of you team doesn't have the necessary training, conduct a training class immediately before proceeding. In fact, given the rapidly changing nature of commercial software tools, remedial training has great benefits.

Though we may often wish to avoid the upfront cost in both time and money, ignoring the need for training is shortsighted. No matter how well trained a team is, unless they've been working together for a long time (a near impossibility in today's environment of mergers and takeovers), the customer will benefit tremendously by insisting that training is conducted immediately on project start-up. The savings in labor and enhancement in the quality of the product will more than pay for this training, though your organization must do its part by having a strong training program, either internal or out-sourced.

For C4ISR Architecture projects that use the Vitech methodology and tool, we have

© 2002 Systems and Proposal Engineering Company. All Rights Reserved.

found it extremely beneficial to conduct special training classes for the team. Vitech has demonstrated its willingness and capability to provide tailored on-site training to projects and proposals on short notice. As a result, the team is better capable to use the methodology and tool correctly, and this also builds the team into a coherent unit ready to attack the problem.

Try to get your customer to understand the need for job-specific training, in addition to your company's standard training program. Include your customer in this training, since it will help you both to communicate better. Then the team becomes more than just the contractor; the team becomes a stronger customer-contractor team, resulting in greater customer satisfaction.

DEFENDING YOUR ARCHITECTURE

You need to be able to defend the work you do for your customer, but more often than not in the C4ISR domain, your customer will provide reports and briefings to a variety of stakeholders, most of whom will have no idea (or interest) in system engineering. This problem has always been present in system engineering. We work very hard to develop logical, structured systems, but few people want to see esoteric IDEF or FFBDs or UML diagrams (and that's the "cool" stuff for us). So how do we communicate, not only with our customer (who may or may not be a good system engineer), but also with our customer's customers?

In our experience, the worst thing you can do is present a bunch of system engineering diagrams as they are properly formulated. You need to have them, but put them into an appendix for the few who will want to examine them in detail.

First, we do high quality system engineering, then we create a simple report that contains the findings of this analysis. However, it is these findings, embodied in key diagrams, developed during system engineering, that need to be presented. But how do we avoid the problem described above? At its simplest, we take the technical results of good systems engineering analysis and translate them into "English." We have found success by converting key diagrams into simple flow diagrams, such as shown in **Figure 4**. This figure shows an enhanced function flow block diagram (EFFBD) from CORE on the left. On the right is the Microsoft PowerPoint rendition of the same diagram. To the purist we have lost important information, but to the general audience we have enhanced understanding significantly.

The Framework products focus on this idea of representing technical details in terms or formats understandable to most decision makers. These formats include the OV-1 (High-level Operational Concept Description), OV-4 (Organizational Relationships Chart), SV-8 (System Evolution Description), SV-9 (System Technology Forecast), TV-1 (Technical Architecture Profile), and TV-2 (Standards Technology Forecast). These products can be easily briefed to almost any audience.

So by focusing in on the findings, then providing appendices with descriptions of the methodology, tools, process, people (short biographies), and detailed results of your system engineering, you provide the customer with the best of both worlds: 1) a simple to understand, get-to-the-point, description of the results; and 2) a detailed, comprehensive report of the work performed (helping justify why they spent all that money).

As a result of applying this approach to your next C4ISR Architecture development project, you will have must greater success, be more competitive and have a more satisfied, better-informed customer.

© 2002 Systems and Proposal Engineering Company. All Rights Reserved.

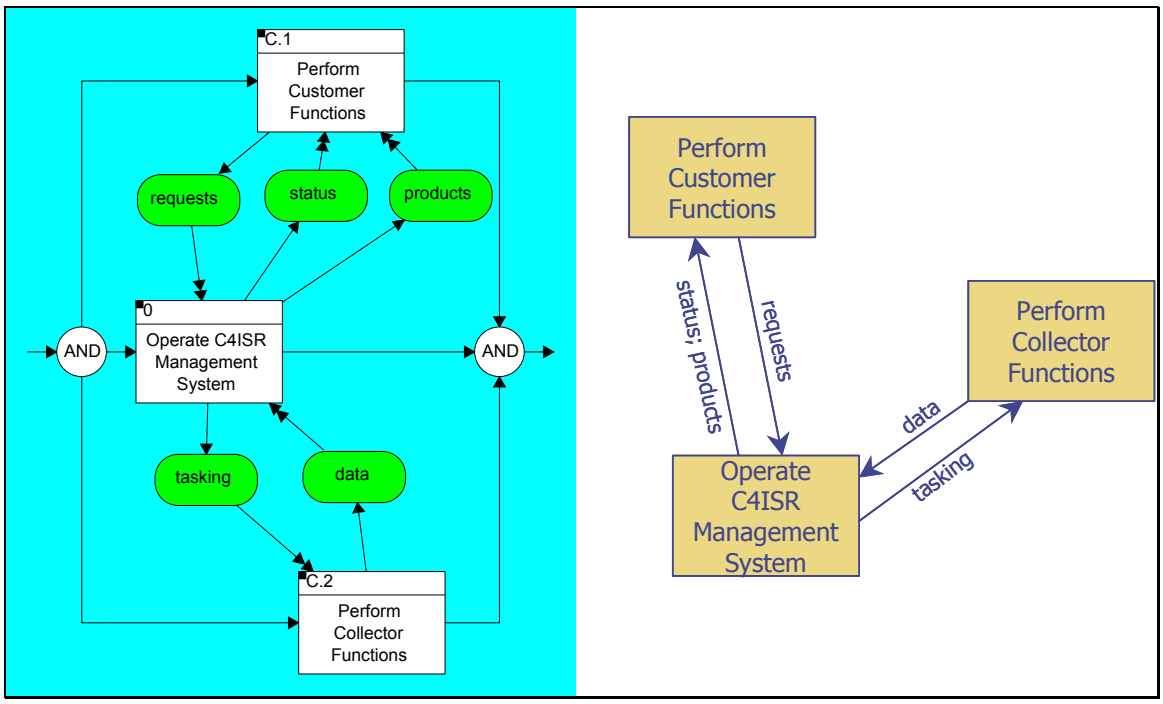

Figure 4. Defending Your Architecture Takes "Viewgraph Engineering" Too

ABOUT THE AUTHORS

Dr. Steven H. Dam is the President of the Systems and Proposal Engineering Company and was a participant in the development of C4ISR Architecture Framework, Version 2.0. He has been a performing system engineer, architect and software developer since the late 1970s.

Mr. James D. Willis, Jr. is a practicing systems engineer. He is also a retired US Air Force Intelligence officer with nearly 30 years experience in defining, acquiring, or using DoD intelligence systems and applying the principles of defensible architecture development.

REFERENCES

[1] C4ISR Architecture Framework, Version 2.0, see www.C3i.osd.mil.org/cio/i3/awg_digitial_library/pfdocs/fq.pdf

[2] OSD Memo, February 23. 1998, Subject: Strategic Direction for a DoD Architecture Framework, (signed USDA&T, Acting ASD/C3I, and Director, JCS/J4).

[3] DeMarco, T., *Structured Analysis and System Specification*, Prentice-Hall, Englewood Cliffs, N.J., 1979

[4] Booch, G., *Object-Oriented Analysis and Design with Applications (2nd Edition)*, Benjamin Cummings, Redwood City, 1993

[5] For more information on UML, see www.rational.com/uml/index.jsp

[6] "Applied Systems Engineering with CORE (introductory course)," VITECH Corporation (www.vitechcorp.com)

[7] For more information on the Zachman Framework, see www.zifa.com/framework2.htm.

© 2002 Systems and Proposal Engineering Company. All Rights Reserved.