2000

A Systems Engineering Methodology for Information Systems

Osmundson, John S.

Systems Engineering, Volume 3, No. 2, 2000
http://hdl.handle.net/10945/44174

Regular Paper

A Systems Engineering Methodology for Information Systems

John S. Osmundson

Naval Postgraduate School, 1 University Circle, Monterey, CA 93943-5000

Received September 3, 1999; Accepted January 28, 2000

ABSTRACT

Complex information systems are often developed without systematic consideration of architectural alternatives partially because systems engineers have lacked a methodology for performing quantitative trade studies of networked systems of sensors, processors, and communications systems. In this paper an approach is discussed for analyzing time-critical information systems and performing systems trades. Information systems are described in terms of design factors with discrete factor levels. Object-oriented models are constructed of the information systems and simulations are run to obtain system measures of performance. Design of experiments is used to drastically reduce the number of models required. The approach is illustrated for an example combat identification information system. © 2000 John Wiley & Sons, Inc. Syst Eng 3: 68–81, 2000

1. INTRODUCTION

A strength of systems engineering is the ability to analyze complex systems problems in terms of fundamental parameters, formulate alternate architectural solutions, perform trade-off analyses of the alternate solutions, and select a best solution based on a reasonable set of selection criteria. Development of credible alternatives and selection criteria has helped system acquirers make better informed acquisition decisions. Trade study methodology has been applied successfully to a large number of aerospace systems and has recently been applied to an increasing number of systems outside of the aerospace industry.

Application of similar systems engineering principles has been lacking in the area of information systems, however, especially in the design and analysis of complex, time-critical networked, distributed information systems, including military command, control, communications, computer, intelligence, surveillance, and reconnaissance (C41SR) systems. Approximately 15 years ago the author had the opportunity to discuss the systems engineering of the World Wide Military Command and Control System (WWMCCS) with a senior scientist at the then Defense Communications Agency (DCA.) At that time the DCA scientist made two points: (1) There was no systematic way to perform trade studies on WWMCCS because tools and method-

Systems Engineering, Vol. 3, No. 2, 2000

ologies were lacking; and (2) it didn't matter anyway because WWMCCS would never change due to the sunk system costs. Today, of course, WWMCCS no longer exists, having been replaced by an even more complex system, but the problem of a lack of any system engineering methodology applicable to large, distributed information systems persists.

Some people still argue that it is not necessary to do trade studies of information systems architectures because often expensive communications systems infrastructures exist that are economically unfeasible to change. This argument violates the systems engineering precept that it is useful to know what an unconstrained solution to a problem is as well as the potential trades in performance and cost from constrained and unconstrained solutions.

Currently in both the commercial and military sectors there is an accelerating trend toward increasing reliance on information systems. Information systems are seen as a way to create cost reductions and competitive advantages to commercial organizations [Kupfer, 1993]. A military counterpart to the commercial use of information systems is the concept of the Global Command and Control System (GCCS) [Griffith, Sielski, and Frye, 1998] and network centric warfare [Cebrowski and Garstka, 1998] which seeks to improve war fighting information and gain a military advantage through the use networked sensors, communications, processing, and display systems.

Many military planners view the solution to the problem of C4ISR systems design to be ever increasing bandwidth. Unfortunately, large bandwidth implies very expensive systems, and even if large bandwidths were affordable, the system could easily overwhelm users with excess information, resulting in far less than an optimum C4ISR solution.

Similar problems exist in the non-military world. Increased interest in commercial applications such as video-on-demand require careful engineering of complex networked systems that provide both high quality consumer satisfaction and attractive profit margins.

The lack of any systems engineering methodology applicable to these problems has become a serious deficiency in the face of the current focus on the power of information. There is an increasing need for application of systems engineering principles to the analysis and design of complex, expensive, and performance-critical information systems.

2. THE CHALLENGE OF INFORMATION SYSTEMS ANALYSIS

The first challenge of systems engineers is to derive requirements for information systems. A number of classical techniques have been developed and that can aid in the analysis of requirements of complex information systems and software systems including IDEF [Moore, Genoese-Zerci, and Sarsi, 1988], data flow diagramming [Yourdon and Constantine, 1978; Ward and Mellor, 1985], data structure analysis [Warnier, 1974; Orr, 1977], and object-oriented analysis [Coad and Yourdon, 1990; Booch, 1994], but each of these techniques has limitations in their ability to quantify performance of system options. For example, in order to quantify data latency in an information system the analysis technique must explicitly express the system in terms of variables, or design factors, that represent time delay elements that can be summed to give an overall system response time. While some of the classical techniques such as data flow diagramming and object oriented analysis do express systems in terms of elements that can sometimes directly or indirectly be expressed as time delays, none of these techniques easily allow the summing of time delay elements, nor are they easily amenable to performing system trades based on quantifiable measures of system timelines. As a result, in the past, system solutions for complex information systems have usually been developed without extensive trade studies and without a solid understanding of system sensitivities to design factors and system input variations.

The most important measure of performance of networks is usually is the time delay from message transmission until message receipt. Message delay has many components including propagation and transmission delays due to physical link properties, processing delays due to message handling processes, and queuing delays due to competition among multiple messages for network resources. Queuing delays are the most important component and the most difficult to analyze [Bertsekas and Gallager, 1991]. Realistic solutions of queuing delays in complex networks has proven to be analytically intractable due to the large number of possible network paths, large number of possible messages and message types to be sent over a typical network, and the large possible number of conditions under which the network might operate. The difficulty of the problem of network analysis is compounded for high bandwidth and time critical systems where detailed timing interface issues are critical to system performance.

Often the system engineer's role is to resolve high-level architectural issues while design engineers resolve issues at lower levels of system detail. A robust systems engineering methodology, therefore, must allow for a layered approach to information system analysis; top-level trades must be addressed first, before lower level trades studies are undertaken. The systems

engineer is most often addressing network performance at the message level, rather than packet or bit level of detail. The systems engineer may also use aggregation to represent a large number of network users by a smaller set of users with higher message transmission rates. Also, since systems engineers are traditionally responsible for carrying out high-level trade studies a systems engineering must represent information systems in a modular manner so that new system configurations can be constructed by efficiently rearranging and reconnecting modular elements of the system.

3. A PROPOSED SYSTEMS ENGINEERING METHODOLOGY FOR INFORMATION SYSTEMS ANALYSIS

The systems engineering methodology described in this paper is based on representing an information system's logical and physical structure graphically and then directly relating the graphical view to an object-oriented system model. System design drivers are identified and variations in the design drivers are represented in alternative system models. Simulations are run to obtain measures of system performance and to determine the best system alternatives.

Information systems are designed to get the right information with the required accuracy to the right recipients within a required timeline. The systems engineer must determine what are the "right" information items, who are the "right" suppliers and recipients of the information, and what are the information accuracy and timeline requirements. A starting point that is familiar to both systems engineers and software engineers is the use of scenario-based analysis. The data structured systems development (DSSD) [Warnier, 1981] methodology, for example, prescribes identifying the producers and consumers of information in a given scenario and then determining the flow of information items between the producers and consumers. Unified modeling language (UML) [Fowler and Scott, 1997] utilizes use cases and sequence diagrams among other constructs to help determine information producers and recipients. These software engineering techniques help the analyst develop a logical view of an information system—a view that emphasizes the information flow and interconnections.

Once a logical view of the system has been constructed it is important to produce a physical view of the system. At this point the needs of systems engineers diverge slightly from the needs of software engineers. Systems engineers are concerned with optimizing the total system consisting of hardware and software, and in a distributed system the hardware elements can include extensive communications systems as well as processors. In addition, optimization of a distributed system is usually equivalent to minimizing system response times since information accuracy is often a function of information timeliness.

An approach to developing a physical view of the system that gives insight into system time behavior is to develop a graphical, thread-based system representation that is similar to UML sequence and swim lane diagrams. An example diagram is shown in Figure 1. Physical elements of a distributed system are arranged vertically on a two-dimensional plot. The physical elements are separated by horizontal boundaries on the plot. Functions performed by each physical element are arranged vertically within the element boundaries according to flow of information between functions, with flow going from top to bottom. Functions are also arranged horizontally for each physical element in terms of time behavior with time progressing toward the right. Thus functions are arranged left to right in the order in which they are performed.

There is no feedback on the plot. If functions are performed more than once as in a feedback loop, the functions are repeated on the plot in an appropriate position further along to the right on the plot. Figure 1 illustrates this graphical method of describing a distributed system for a generic system consisting of physical elements *A*, *B*, and *C*, and functions *F*l, *F*2,… within each physical system. The methodology works equally well for an object-oriented analysis (OOA) and design (OOD) of a distributed system.

Separate physical elements are often linked by external communications systems in a typical distributed system. The external communications systems might be wire, fiber-optic, radio, or satellite links, for example. Also, there are internal communications between functions within a physical element. If the physical element is a computer these would correspond to interprocess communications. For object-oriented systems, there can be interobject messaging, which effectively adds further communications overhead to the system.

A useful technique to aid in understanding system time behavior is to construct functional event threads for scenarios of interest. For example, if an event triggers execution of function *F*1, in system *B* in Figure 1 which in turn triggers function *F*4 in system *A*, and then a succession of other functions in systems *A*, *B*, and *C* before ending in a terminating function, the time-ordered series of functions is referred to as a functional thread. An example functional thread for the generic system is illustrated in Figure 1, where the functions invoked by the thread are linked by a directed arrow.

Time delays are associated with performance of each function and with each communications process.

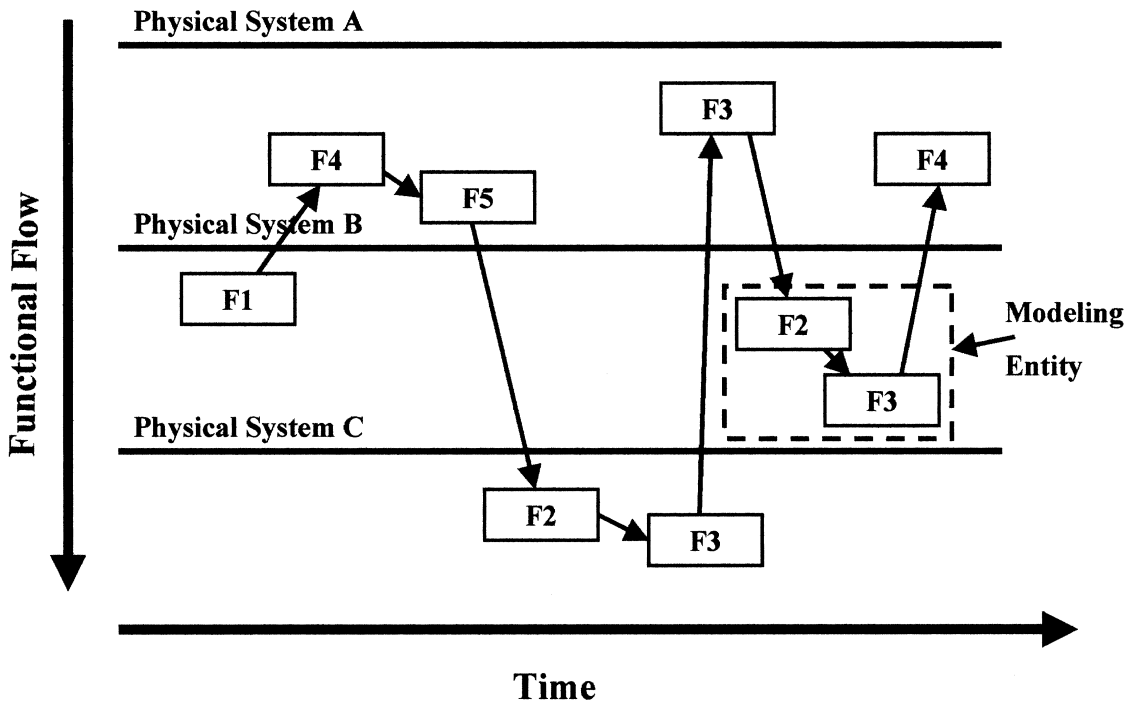

Figure 1. A graphical method of describing a distributed system.

One possibility for optimizing system performance is to minimize the total time delay associated with each thread. Since there can be many threads (thousands of threads for complex systems), some of the candidate objectives of system optimization are to minimize the average thread time, minimize the maximum thread time, minimize the average time of the threads associated with the most probable events, etc.

Data latency is due to network access methods, message queuing, processing delay, network capacity, and propagation path delays. It is impractical to attempt to model these delays and analytically solve the resulting equations for system threads. Also it is unlikely that optimization methods of operations research can be applied because, in general, the functional forms of the time delays are not known. A more practical approach is to model a distributed system using network simulation tools. Thread time delays can be obtained from simulation runs. The system architecture can be varied by rearranging the model and new thread time delays obtained. The process can be repeated until an optimal or near-optimal system performance is achieved.

4. APPROACH TO COMPUTER MODELING

Modern network design and analysis software applications have the capability to model information systems analogously to the method described above. Network elements can be described as icons having the properties of queues, delays, routers, switches, combiners, and other delay elements. The icons can be grouped to represent functional properties of a network or, as appropriate, network functions can be grouped and represented by a single modeling icon as shown in Figure 1. These icons can be graphically linked to form models of physically distributed systems. Event generators are built in functions and can be programmed to trigger the creation of messages and resultant threads in a network model.

Some special purpose network modeling applications tools such as the U.S. Army's Network Assessment Model [Mallette and Copeland, 1990] have been developed for military use. These applications are usually designed to address very specific issues and often are unique to a given service. Commercial applications by contrast are usually aimed at generic applications and are designed to be extremely flexible in the level of modeling detail. In general commercial tool sets are better suited for systems engineering trade studies.

Several factors are important considerations when selecting a distributed system modeling tool. These include, among other factors: (1) platform requirements; (2) cost of the tool; (3) ease of use; (4) time required to model a system using a given application; (5) accuracy of the modeling application; (6) time

required to run a model on a given platform; (7) maximum size problem that can be treated using the application; and (8) ability of the application to run interactively with other simulations, with hardware in-the-loop, or a combination of other simulations and hardware in-the-loop.

Some examples of modern network design and analysis tools are OPNET, COMNET III, Workbench, and EXTEND. These applications, as well as the application G2, were evaluated in depth by Gebhardt [1997] and a condensed summary of her results is shown in Figure 2. No one tool is optimal for all problems—a network design and analysis application should be selected that is appropriate to a particular problem. However, key conclusions of Gebhardt's evaluation is that the application EXTEND offers a uniquely low cost tool that runs on commonly available personal computers, and is easy to learn and use. Smith, in a separate study [Smith, 1998], modeled identical networks in EXTEND and OPNET and compared detailed results of the two simulations. Smith's results showed excellent agreement between results obtained using the two applications, with most measurements of the same parameters in the two models being within 1% of each other. The accuracy of the EXTEND results are certainly more than sufficient for most high-level systems analyses and the author concludes that EXTEND is often a good choice of application for performing trades of initial network design concepts.

Before developing models the systems engineer must identify the top-level requirements and the appropriate trade space and candidate system options. One of the first steps is to determine why information must flow and what the content of the information must be, including from whom and to whom the information must be passed, and within what timelines. In many situations the system engineer would like to examine the effects of many variables on a dependent variable. The variables are often referred to as design factors, and the dependent variable is often referred to as an objective function. The systems engineer must identify the design factors. Design factors might include the type of arrangement of node interconnections, methods of access, methods of routing and controlling flow of information, and bandwidths of the various links in the network. Typically each of the design factors can be varied (the design factors can be said to have several different states or levels) and combinations of the various design factors in different levels represent potential system options.

Next, models of the system are constructed in a modular manner so that design factors are represented by an association with modeling application objects. System options are represented by rearranging the objects and by varying the object attributes from model to model. The system engineer may find it convenient to aggregate many individual transmitting nodes, or sources of information, into a single node as long as the message traffic of the single node is appropriately scaled upward.

Once system design factors and factor states have been identified, system models must be built that enable all of the design options to be analyzed. Often the total number of options can be high, resulting in a formidable modeling task. For example a full parametric variation of a system involving m design factors each with L levels would result in the need to create N separate system models where $N = L^m$.

	COMNET III	OPNET MODELER	EXTEND	WORKBENCH	G2
General Usage	Network Planning Tool, LAN/WAN	Comm Networks, Computer Systems and Applications	Simulation Tool	System Level Simulation Tool	Intelligent Real Time Systems
Vendor	CACI Products Company	Mil 3, Inc.	Imagine That, Inc.	SES	Gensym Corp.
Platforms	PCs (Windows 95, 98, NT), UNIX, DEC, HP UX, SGI, AIX, Solaris	Sun SPARC, DEC, HP UX, SGI, Windows NT	PCs (Windows 3.1, 95, 98, NT), MacIntosh	Sun Micro-stations, IBM, HP	Digital, IBM, HP, Sun Micro-stations, PCs (Windows 95, 98, NT)
Approximate Price	$48,500	$18,000	$695	$23,500+	$40,800

Figure 2. A comparison of selected network design and analysis tools [Gebhardt, 1997].

5. APPLICATION OF DESIGN OF EXPERIMENTS

One approach to reducing the initial system modeling task to one of reasonable size is to use orthogonal arrays to identify the models required to cover the variable or design factor space and give sufficient information to identify the system arrangement that optimizes the dependent variable or objective function. This approach works well if the information system can be optimized by optimizing a single objective function such as data latency. The concept of orthogonal arrays originated with the work of Fisher [1948] and has been the subject of numerous studies of the design of experiments [McLean and Anderson, 1984] and the statistics of design of experiments [Raghavarao, 1971]. Taguchi [1987] gives a treatment of orthogonal arrays that emphasizes practical applications as opposed to mathematical analysis and is very accessible for engineers.

As a simplified example of orthogonal arrays consider a system that has three design factors each with two possible levels. A full parametric variation would require eight models to be built and run in order to determine an optimum configuration. The orthogonal array for this case is shown in Figure 3. Here the eight models that are required using full parametric variation arc reduced to four models using design of experiments. The first model is built with all of the three design factors in their first level. The second model is built with the first design factor in its first level and the other two factors in their second level, and so forth. A characteristic of an orthogonal array is any two columns of the array contain all possible combinations of states of different factors the same number of times.

Results of the models are analyzed as follows: The values of the objective function for all runs obtained with models with design factor #1 in level 2 are added and divided by the number of runs with design factor #1 in level 2, and the same procedure is followed for all runs obtained with design factor #1 in level 1. Let $R(F_1a,F_2b,F_3c)$ be the result of a simulation run of a model having design factor 1 at level a (F_1a), design factor 2 at level b (F_2b), and design factor 3 at level c (F_3c.) The average result for simulation runs for models shown on Figure 3 with design factor 1 in level 1 is given by

$$\{R(F_1 1, F_2 1, F_3 1) + R(F_1 1, F_2 2, F_3 2)\}/2,$$

and the average result for simulation runs for models with design factor 1 in level 2 is given by

$$\{R(F_1 2, F_2 1, F_3 2) + R(F_1 2, F_2 2, F_3 1)\}/2.$$

In each case the results for design factor F_1 in a given state include the effects of design factors F_2 and F_3 equally in all of their possible levels. Thus the results for design factor F_1 are independent of (or orthogonal to) the effects of F_2 and F_3 which is a characteristic of orthogonal arrays. This does not mean that the design factors are physically independent, but only that a mathematical analysis will produce independent results for each design factor [Barker, 1985.] This characteristic of orthogonal arrays allows the effect of each design factor to be determined separately by analyzing all of the simulation runs as a unit. None of the four models shown on Figure 3 is necessarily an optimum arrangement of design factors and levels. However, analysis of the results of an entire set of simulation runs will show the dependence of the objective function on the choice of levels for each of the design factors, indicating the arrangement of design factors and levels that will yield an optimum system.

Simulations	Design Factors		
	Factor 1	Factor 2	Factor 3
Model 1	Level 1	Level 1	Level 1
Model 2	Level 1	Level 2	Level 2
Model 3	Level 2	Level 1	Level 2
Model 4	Level 2	Level 2	Level 1

Figure 3. Orthogonal array for a system with three design factors each of which can have two levels.

The reduction from eight cases to four cases is not dramatic but for larger dimension systems the reduction in modeling effort can be substantial. A system with seven design factors each with two states would require 128 models using full parametric variation but only eight models using design of experiments.

An orthogonal array must be constructed that is appropriate to the problem of interest, depending on the number of design factors present, the number of possible levels for each design factor, and whether any of the design factors interact with one another. A number of orthogonal arrays, known as standard orthogonal arrays, have been developed by previous researchers [Taguchi and Konishi, 1987], and other arrays can be constructed by applying rules to standard arrays. Roy [1990] gives an excellent discussion on the construction of orthogonal arrays by applying design rules to standard orthogonal arrays.

A distributed information system must be described in terms of design factors that have discrete parameter levels in order to apply design of experiments. The systems engineer must analyze the problem and determine an appropriate set of design factors and levels for each factor. One design factor might be method of network access and associated choices of discrete design levels might be time division multiple access (TDMA) or code division multiple access (CDMA). Other design factors might have continuous variations rather than discrete levels. Continuous design factors, however, can often be quantized into discrete levels. Bandwidth of various links in an information system is often a design factor. Discrete levels of bandwidth can be chosen and simulation results used to indicate the relative improvement of a high level of bandwidth compared to moderate or low levels of bandwidth.

6. ILLUSTRATIVE EXAMPLE

The systems engineering methodology described above is illustrated by applying the methodology to a current military distributed information problem. The movement toward digitization of the battlefield and the concept of network centric warfare leads to many complex military network design and analysis problems. Among these problem areas is the compelling need to provide combat identification (CID) for large, joint force operations. CID solutions for future forces often propose equipping all, or a large number, of friendly combat force elements with situational awareness sensors—usually Global Positioning System (GPS) based position indicators—as well as ancillary sensors for identifying other forces [Marshall Associates, 1999]. The data resulting from the CID sensors would then be distributed to all required users through a networked information system. The most important measure of effectiveness (MOE) of a distributed information CID system is often data latency, i.e., the time required to get CID sensor data to required users under a variety of battle conditions, and from a systems engineering perspective it is important to understand how data latency depends on system design alternative. Data accuracy is also important in a CID system. However, data accuracy also depends on data latency as well as the type of CID sensors employed. The choice of CID sensors is a separate system engineering issue that will not be addressed here.

6.1. Battle Scenario

Following the proposed methodology we begin by constructing an example joint forces operational scenario for which forces and a sequence of force movements and supporting fires are specified. A littoral joint forces small scale amphibious operation scenario [Combat Systems Report, 1997] was constructed as a means of identifying CID requirements. The operation consisted of approximately 5000 blue (friendly) force entities. Blue forces were defined to consist of specific types, numbers, and locations of dismounted troops, artillery and mortars, various other ground forces, mobile vehicles, fixed and rotary wing aircraft, landing craft, and offshore naval units. Dismounted troops were aggregated into squads and platoons. Each of the blue force entities was further defined in terms of weapons types, maximum speeds, and other characteristics. A countering red (enemy) force was also defined in the same manner. The scenario was run as a combat simulation using JANUS [1993], a force-on-force simulation system. A view from the JANUS simulation of the scenario is shown in Figure 4. At the time shown in Figure 4 blue forces had occupied areas near a port and an airfield and were preparing to link up with each other. Red forces occupied the area slightly to the south and between the airfield and port. The blue forces were concentrated in three areas as shown in Figure 4, and for purposes of analysis the forces were assumed to be equally distributed among the three areas.

Locations and interactions of the blue and red forces changed during the JANUS simulation run. As blue force entities came into weapons range of one another, opportunities for fratricide existed. At times during the simulated battle there were critical periods when various blue force entities quickly came in close proximity to each other, creating a need for fast and accurate transmittal of CID information.

A simple graphical logical and physical view of the CID problem is shown in Figure 5. Information from

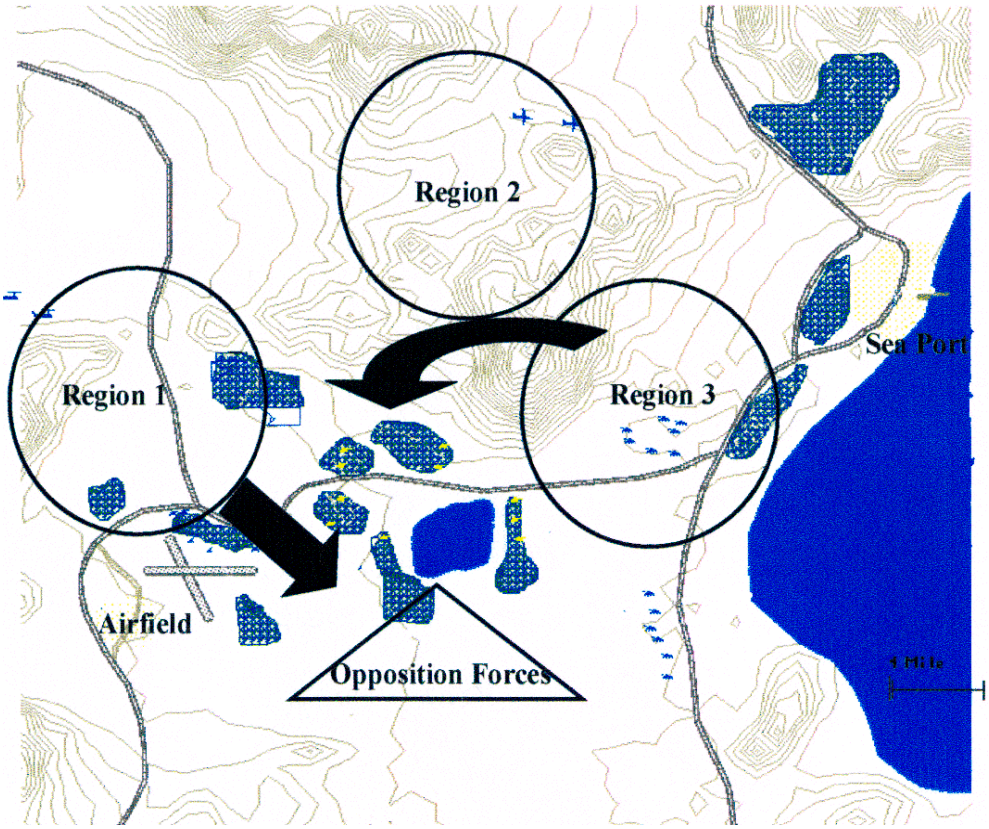

Figure 4. Joint amphibious operation scenario.

CID sensors must be transmitted to the shooters in a timely manner. For simplicity the large number of distributed systems on the battlefield are shown as systems *A*, *B*, and *C* in Figure 5. Processing can be done at the sensor, at the shooter, or at intermediate nodes. Sensors can be both organic to shooter platforms or remote. The CID information system must link the

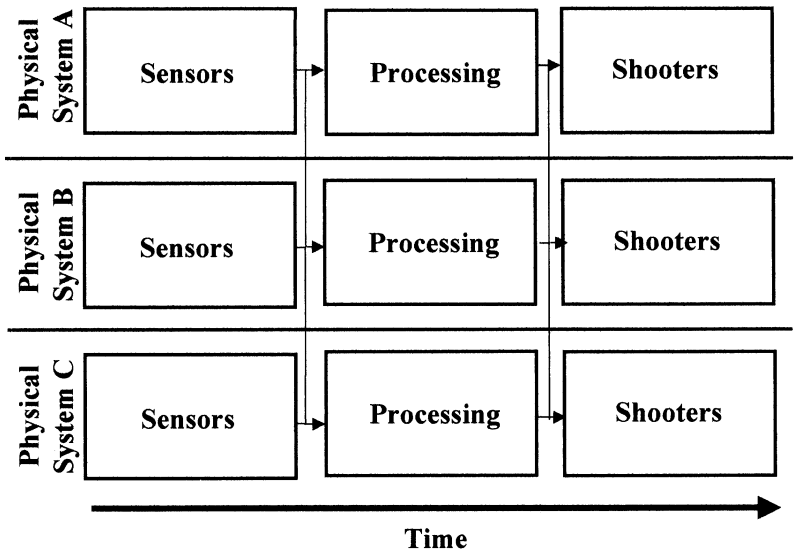

Figure 5. Graphical view of combat identification problem.

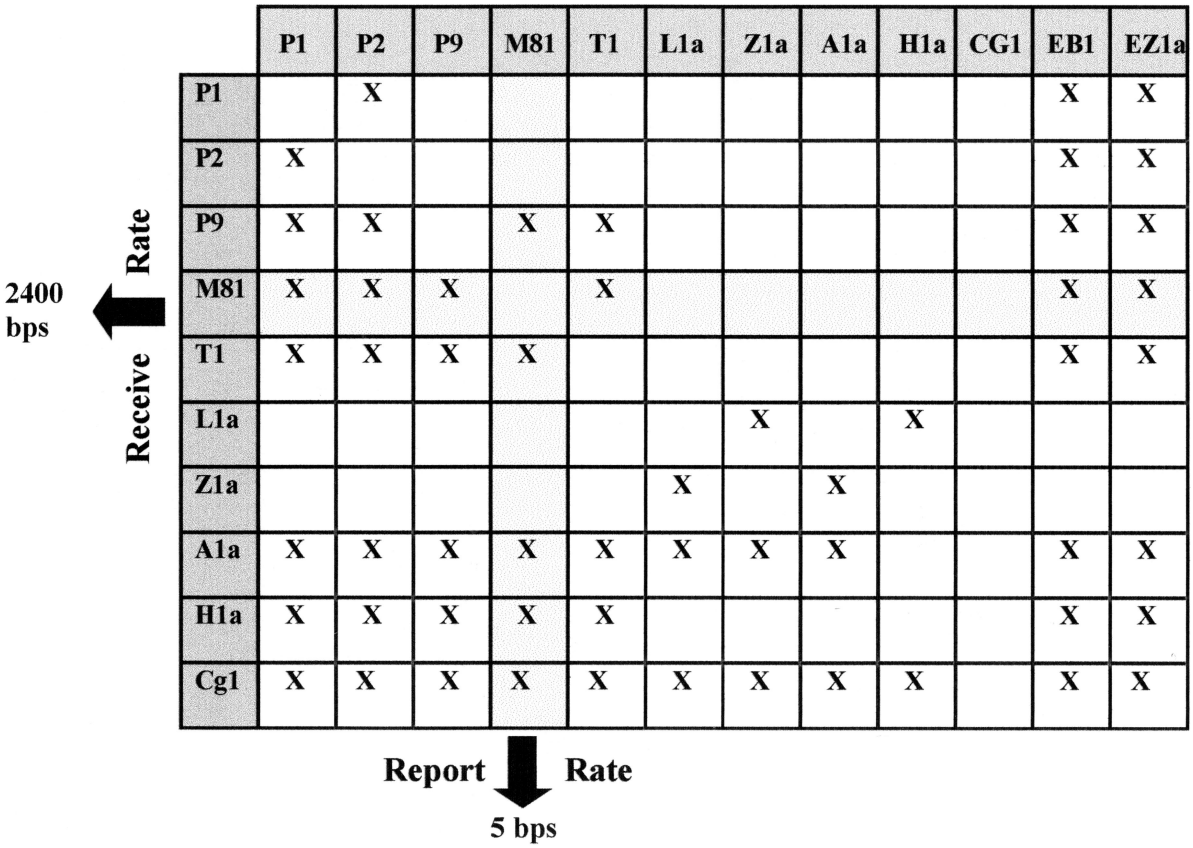

Figure 6. A portion of the force interaction matrix midway through the scenario. Labels at the top of the columns and to the left of the rows correspond to the force elements in the combat scenario. M81, for example, corresponds to an 81 mm mortar.

required sensor information to shooters within required timelines.

Information from the JANUS simulation was used to construct a force interaction state matrix [Melich and Osmundson, 1995] for the scenario. The force interaction state matrix lists all blue force entities along the *x*- and *y*-axes of a two-dimensional plot. All possible lethal interactions between blue force entities are indicated at the intersection of the appropriate force entities on the two-dimensional plot, resulting in a force interaction matrix. Since the interactions change during the battle, several snapshots of the interaction matrix were developed to determine average and worst case conditions from a CID perspective. Details about the interactions, including interaction dynamics, were derived from the weapons and motion characteristics of each force entity. A portion of the interaction matrix midway through the simulation is shown in Figure 6.

Notations along the *x*- and *y*-axes of Figure 6 refer to the specific force elements in the JANUS scenario; for example, M81 refers to an 81 mm mortar unit. Information reporting and receiving rates can be determined for each of the elements on the interaction matrix by determining their relative locations during the scenario as well as their weapons characteristics. For example the M81 is shown to have a report rate of 5 bps and a receive rate of 2400 bps at the time represented by Figure 6.

CID data rate requirements were derived from the interaction matrices by assuming a generic sensor reporting position and ID, at a minimum, for each of the battlefield entities. A message size of 500 bits was assumed to be required in order to transmit position and ID information about each entity, including message encryption. Reporting rates were determined to fall into three main categories: (1) The reporting rate for dismounted troops and a background reporting rate for all entities for overall situational awareness was determined to be approximately 5 bps; (2) the reporting rate for mobile ground vehicles was approximately 50 bps; and (3) the reporting rate for fast moving entities, including rotary and fixed wing aircraft, was approximately 500 bps.

6.2. Candidate Network Architectures

When situational awareness sensors with all processing onboard the sensors are used, the problem illustrated by Figure 5 becomes one of finding the best distributed communication architecture to support CID information needs. The interaction matrices suggest various approaches to CID network architectures. We note that forces are often grouped by geographical location, and/or by type of force element. This suggests network architectures that connect information sources and information users based on geographical areas, or type of force element, or a combination of both geographical location and type of force element.

Time evolving interaction matrices also show that forces need to be connected across geographical areas and across types of force elements according to information needs, and that these information needs change dynamically depending on the situation. Additionally, the requirements for connection paths are not necessarily symmetric. Some suppliers of CID information may have little need for CID from the rest of the CID system while other users of information may require large amounts of CID data.

We begin our analysis of CID architectures by considering major design alternatives or design factors and levels for a CID network. In order to simplify the problem of analyzing alternative network architectures we restrict ourselves to considering five major network design factors: (1) grouping of CID information suppliers and users; (2) bandwidth of the CID network; (3) degree to which smart push of information is implemented; (4) method of network access; and (5) dissemination network architecture. Figure 7 shows the five major options considered in this study in terms of network subelements and function options. The first column in Figure 7 lists the groupings and subgroupings of CID information suppliers into local information reporting subnetworks that were chosen for analysis.

Each subnet can connect entities in a ring or star pattern, but, for the purposes of this study, star patterns are assumed. Inter- and intranetwork access was restricted to time division multiple access (TDMA) and to "bandwidth on demand" or Asynchronous Transfer Mode (ATM)-like access. These two choices bracket existing tactical military communications technology and a very advanced network access technology. The overall network can be designed with or without smart push of information. One or more nodes that have total situational awareness including knowledge of all of the friendly force entity characteristics can maintain a dynamic interaction matrix of the form shown in Figure 6. If smart push of information is implemented, it can be used to transmit time-critical CID messages with higher priority and high frequency while limiting background traffic to lower rates. This effectively reduces the utilization of link bandwidth. In addition, the bandwidth of each network is considered to be a design variable, with bandwidth restricted to three levels: low—9600 b/s, medium—28.8 kb/s, and high—115 kb/s. These levels correspond to typical military tactical link bandwidths and current demonstration CID sensor system bandwidths. Asymmetric architectures are also considered, where reporting of CID information is forwarded on tactical links and disseminated using global broadcasting system (GBS) or global multicasting system from satellites, aerostats, or airborne nodes. For these cases the bandwidth of the dissemination system is assumed to be 3 Mb/s, much higher than typical tactical reporting links. Broadcast CID information

Top-Level Combat Identification System Design Factors

Design Factor Levels	Reporting Grouping	Bandwidth	Smart Push	Network Access	Dissemination Method
	Type	9.6 kbps	Yes	TDMA	Symmetric
	Area	28.8 kbps	No	Access on Demand	Asymmetric
	Type & Area	115 kbps			

Figure 7. Combat identification system network design factors and factor levels.

Combat Identification System Design Factors

Simulations	Access Type	Smart Push	Dissemination Method	Grouping	Bandwidth
Model 1	TDMA	Yes	Symmetric	Type	9.6 kbps
Model 2	TDMA	Yes	Symmetric	Type & Area	115 kbps
Model 3	TDMA	No	Asymmetric	Type	115 kbps
Model 4	TDMA	No	Asymmetric	Area	9.6 kbps
Model 5	Access on Demand	Yes	Asymmetric	Area	28.8 kbps
Model 6	Access on Demand	Yes	Asymmetric	Type & Area	28.8 kbps
Model 7	Access on Demand	No	Symmetric	Area	115 kbps
Model 8	Access on Demand	No	Symmetric	Type & Area	28.8 kbps

Figure 8. Combat identification models.

might have to be filtered at each user's receiver in order to prevent information overload of the users. An alternative would be to multicast return information by adaptively forming packets of CID information based on geographic area or unit type with packets headers for cueing of user's receivers.

Options shown in Figure 7 lead to over 70 possible CID network architectures based on the total number of combinations of local groupings and function choices. There are potentially many measures of effectiveness of a network, but the first measure should be that the architecture meets the timeliness requirements of the mission. Other important considerations include information accuracy, network robustness (vulnerability to increases in the network load, sensitivity to the environment, and susceptibility to network failures and outside attacks are some of issues associated with robustness) and cost.

6.3. Network Simulations

Network features, such as those listed on the column headings of Figure 7, can be encapsulated as object models and as connections of object models and represented graphically using modern object-oriented simulation tools. System options can be modeled by graphically manipulating individual object models, rather than rewriting code. Thus, many architectural variations can be readily modeled and simulated in a reasonable length of time. Design of experiments methodology can be used to reduce the total network architecture trade space to a reasonable level.

The more than 70 CID architectures corresponding to a full parametric variation of the design factor choices in Figure 7 were reduced to a set of eight different combinations of design parameter choices, shown in Figure 8, that were modeled using EXTEND. These choices of subnetworks and functions are a partial, but representative set of the total set of architectures required to be modeled, based on the use of orthogonal arrays and design of experiments methodology, in order to obtain a near optimum architectural solution.

Figure 9 shows a top-level view of a representative EXTEND model for an architecture incorporating TDMA access, asymmetric (GBS) information dissemination, and smart push of information. The network in Figure 9 and all other network models consist of three subnetworks, with three nodes on each subnetwork. Each node can be modeled to represent a large number of information suppliers and information users by appropriate scaling the message loads at each node to correspond to the simulation of the scenario shown in Figure 4 and the resultant CID information flows. Many nodes in Figure 9 are hierarchical blocks so that the details of the node model extend several layers deeper. Bandwidths and reporting entity groupings are determined by parameter settings within the model. Timers were inserted in every model to measure the delay from generation of information at a several dif-

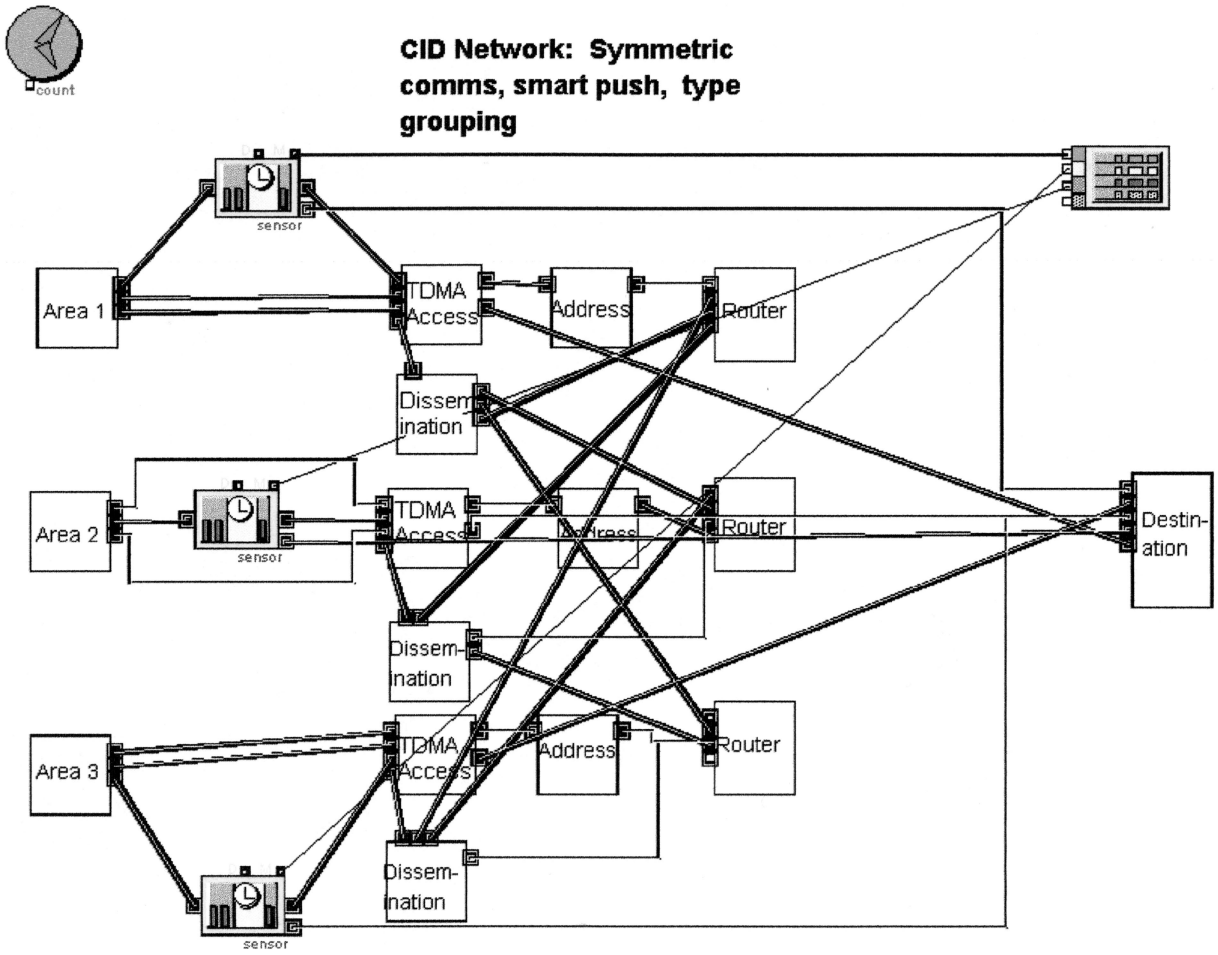

Figure 9. Object-oriented model of CID network.

ferent reporting nodes to receipt of the information by several different receiving nodes. Measurement principles of complex network models is still an area for research, but the author follows the principle of measuring network models at inner and outer subnetworks. For the purposes of this analysis, architectures were compared based on the delays encountered by high priority information originating at the first node of the first subnetwork and the third node of the third subnetwork in reaching their destinations.

Results of computer simulation runs for each of the design factors listed in Figure 8 are shown in Figure 10. The results are plotted in terms of relative delays of critical information generated at the first node of subnetwork 1 (referred to as subnet 1 in Fig. 10) and subnetwork 3 (subnet 3 in Fig. 10) reaching the user of the information for the eight selected network models. Models 1–4 shown in Figure 8 all have access type TDMA. Relative delays obtained by running models 1–4 were averaged to obtain the result corresponding to TDMA access in Figure 10. Results from running models 5–8 were averaged to obtain the result corresponding to access by demand in Figure 10. The two points corresponding to TDMA and demand access were then connected by a straight line to visualize the difference between access types. Other results were obtained in a similar manner.

The first set of data in Figure 10 shows the effect of varying the bandwidth from 9.6 to 28.8 to 115 kb/s. As expected, network delays are reduced with increasing bandwidth, but not dramatically so in going from 28.8 to 115 kb/s. The next set of data shows the effect of disseminating information using the same return paths as for reporting—a symmetric network—or disseminating information using an asymmetric architecture, namely, a GBS system. The improvement in using asymmetric dissemination is pronounced. The third set of data in Figure 10 shows the effect of introducing smart push. Again, definite improvement can be obtained by utilizing smart push of information, and this can sometimes result in a bigger improvement in network performance than increasing bandwidth. The

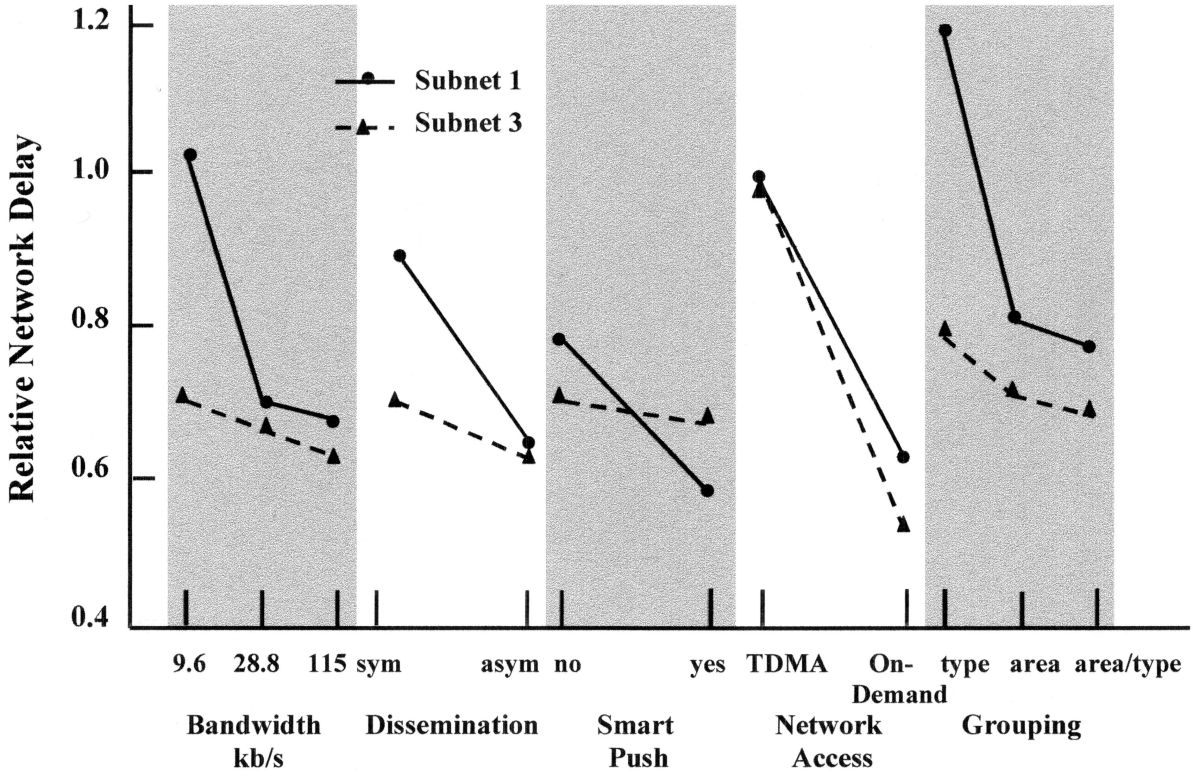

Figure 10. Results of simulation runs.

fourth set of data relates to network access. Access by demand shows dramatic improvement over conventional military tactical data network TDMA methods. The fifth set of data shows the effect of grouping force elements for reporting and dissemination purposes by function, by geographical area, or by function within a given geographical area. A significant improvement is shown by grouping force elements by geographical area and a further smaller improvement is shown by grouping force elements by both type and area.

While Figure 10 shows results in terms of relative delays the EXTEND models give results in absolute terms that are as accurate as the summed effects of the estimated component delays entered into the model. Absolute results could be compared to CID timeline requirements established from JANUS simulations in order to identify those modeled system options that met requirements. The set of system options meeting requirements could be further quantified in terms of cost, risk or other system parameters to form the basis of system trades.

The models would need to be tested against a wide range of scenarios and battle conditions, and scaled to larger and smaller conflicts before the results could be claimed to apply universally, but the results shown in Figure 10 do give insight into the nature of the CID problem and network centric warfare. Increased bandwidth is important, but in this particular example further increases in bandwidth above 28.8 kbps are much less important than asymmetric dissemination, implementing smart push of information, providing access on demand, and grouping entities by geographical region.

Further steps in the analysis procedure would be to consider the interaction of CID with other battlefield functions and to consider additional design factors. For example, CID should be incorporated into an overarching information system that would transmit remote firing and targeting data as well as other information. Reliability and survivability of networks would need to be analyzed.

7. CONCLUSION

The methodology presented in this paper has described a systems engineering approach to the problem of understanding architectural trade-offs of complex, time-critical information systems. Information systems requirements and network architectures can be established for complex, highly dynamic systems by developing scenarios, identifying system design factors and system alternatives and then modeling and simulating the alternative configurations. Object-oriented model-

ing and simulation, and design of experiments methodologies, provide mechanisms for efficiently formulating and analyzing high-level solutions to information system requirements.

REFERENCES

T.B. Barker, Quality by experimental design, Marcel Dekker, New York, 1985.

D.P. Bertsekas and R. Gallager, Data networks, 2nd ed., Prentice-Hall, Englewood Cliffs, NJ, 1991.

G. Booch, Object-oriented analysis and design, 2nd ed., Benjamin/Cummings, Redwood City, CA, 1994.

A.K. Cebrowski and J.J. Garstka, Network-centric warfare: Its origin and future, U.S. Nav Inst Proc, January 1998, pp. 28–35.

P. Coad and E. Yourdon, Object-oriented analysis, Prentice-Hall, Englewood Cliffs, NJ, 1990.

Combat System Science and Technology SE4021 Project Report—Combat ID, Naval Postgraduate School, Monterey, CA, June 13, 1997.

R.A. Fisher, Statistical methods for research workers, Oliver and Boyd, Edinburgh, 1948.

M. Fowler and K. Scott, UML distilled, Addison-Wesley, Reading, MA, 1997.

H.L. Gebhardt, Communication modulation simulators: An assessment, Masters of Science in Command, Control, Communications and Intelligence Thesis, Naval Postgraduate School, Monteray, CA, June 1997.

J. Griffith, K. Sielski, and H. Frye, C4ISR handbook for integrated planning, Office of the Assistant Secretary of Defense, Arlington, VA, 1998.

JANUS 3.X user's manual, Titan Applications Group, Leavenworth, KS, 1993.

A. Kupfer, The race to rewire America, Fortune 127(8) (April 1993), 42–61.

A.J. Mallette and G.L. Copeland, Advanced technology for command and control systems engineering, AFCEA International Press, Fairfax, VA, 1990, pp 215–226.

Marshall Associates, Combat Identification Study Final Report, USMC Contract No. M67854-97-C-3028, Sterling, VA, 1999.

R.A. McLean and V.L. Anderson, Applied factorial and fractional designs, Marcel Dekker, New York, 1984.

M.E. Melich and J.S. Osmundson, A systems analysis approach to designing evaluations of multiservice combat identification procedures and capabilities, CISC-95, Naval Postgraduate School, Monterey, CA, November 14–16, 1995.

K. Moore, F. Genoese-Zerci, and D.L. Sarsi, IDEF ALPHA user's manual, TR-358-1, ALPHATECH, Burlington, MA, April 1988.

K.T. Orr, Structured systems development, Yourdon Press, New York, 1977.

D. Raghavarao, Constructions and combinatorial problems in design of experiments, Wiley, New York, 1971.

R. Roy, A primer on the Taguchi method, Van Nostrand Reinhold, New York, 1990.

B. Smith, The development of a littoral region area communications network in support of operational maneuver from the sea, Masters of Science in Information Technology Management Thesis, Naval Postgraduate School, Monteray, CA, June 1998.

G. Taguchi, System of experimental design (2 volumes), UNIPUB/Kraus International, Lanham, MD, 1987.

G. Taguchi and S. Konishi, Taguchi methods, orthogonal arrays and linear graphs, ASI (American Supplier Institute) Press, Dearborn, MI, 1987.

P.T. Ward and S.J. Mellor, Structured development for real-time systems, Yourdon Press, New York, 1985.

J.D. Warnier, Logical construction of programs, Van Ostrand Reinhold, New York, 1974.

J.D. Warnier, Logical construction of systems, Van Nostrand Reinhold, New York, 1981.

E.N. Yourdon and L.L. Constantine, Structured design, Prentice-Hall, Englewood Cliffs, NJ, 1978.

John Osmundson is an Associate Professor in the Command, Control and Communications Academic Group at the Naval Postgraduate School in Monterey, CA. His research interests are in applying systems engineering and computer modeling and simulation methodologies to the development of system architectures, performance models, and system trades of time-critical information systems and in software project management. Prior to joining the Naval Postgraduate School in 1995 Dr. Osmundson worked for 23 years at Lockheed Missiles and Space Company in Sunnyvale and Palo Alto, CA as a systems engineer, systems engineering manager and manager of advanced systems studies in the LMSC Research and Development Division. Dr. Osmundson is a charter member of the San Francisco Bay Area chapter of INCOSE and is a member of the IEEE. He received his B.S. in Physics from Stanford University and his Ph.D. in Physics from University of Maryland.