VIRTUAL SYSTEM ACQUISITION: *APPROACH AND TRANSITIONS*

Walt Scacchi and Barry Boehm

There is a pressing need to make software system acquisition more agile and adaptive, through evolutionary modeling, simulation, and development of the system being acquired. Here we'll describe a new vision for the re-tooling and reengineering software system acquisition into a form we call VISTA, denoting an approach to the virtual acquisition of these systems. We will describe this new approach, and then discuss the technical and organizational transitions that must be investigated and managed to ensure the eventual success of such a radical change to software system acquisition.

The acquisition of major software-intensive systems is often problematic. Recent reports from the U.S. General Accounting Office (GAO, 1995; GAO, 1997) describe a number of problems with the way complex systems are acquired. The current acquisition problems include:

- difficulty in establishing viable and cost-effective system requirements;
- overly optimistic cost, schedule, and performance estimates;
- concurrent development and production of systems; and
- commitment to system production before adequate demonstration or testing that determines system viability is completed.

To no one's surprise, modern and future weapon systems increasingly represent software-intensive systems. In addition, the Department of Defense (DoD) and other government agencies rely on the acquisition and use of computer-based information systems to manage their recurring organizational and operational activities. Many of these management information systems are often running on outdated computing platforms that must be replaced or modernized.

The DoD has established acquisition strategies that move it toward commercial acquisition practices. One strategy embodies the idea that the feasibility and ability to produce advanced technologies can often be demonstrated before they are incorporated into acquisition programs. For example, the use of advanced concept

technology demonstrations can more directly involve war fighters and users in demonstrating the operational feasibility of new technologies and concepts before commitments are made to full-scale acquisition. Another strategy rooted in the Defense Acquisition Workforce Improvement Act (DAWIA) establishes benchmarks for a more professional acquisition workforce with defined training and education requirements, and acquisition career paths. The goal of this act is to provide an acquisition workforce that is more responsible for improving program costs and schedule estimates. Finally, in 1994 the Office of the Secretary of Defense began pursuing a strategy to reengineer the systems acquisition review process. This includes an effort to reduce acquisition costs (including overhead costs) through the adoption of business processes characteristic of world-class commercial buyers and suppliers.

The overall way in which the federal government conducts its acquisition practices has been reviewed and redesigned in response to the Federal Acquisition Streamlining Act (FASA) of 1994. Among other things, the FASA requires incentives and a performance-based approach to managing acquisition programs. This emphasizes streamlining the acquisition process and proposes greater reliance on commercial products and processes. Also, concepts for applying commercial practices to DoD software system acquisition have been addressed in Defense Science Board reports.

Thus, we are at a time when there is substantial opportunity to rethink how the acquisition of software-intensive systems should occur to address the recurring problems. At the same time, we should pursue new opportunities to reengineer the systems acquisition process that can realize savings, efficiencies, increased satisfaction, and continuous improvement. Similarly, we should provide a strategy for managing the transition to these reengineered system acquisition processes, as they can represent a radical departure from current practices. Subsequently, we seek to explore how these opportunities can be pursued through use of advanced information processing tools, techniques, and concepts. Our objective is to make the acquisition of software-intensive systems more agile and adaptive. Relevant information technologies include those for:

Walt Scacchi is Director of the ATRIUM Laboratory at the University of Southern California (USC). He has been on the USC faculty since 1981, and is a faculty principal at the USC Center for Software Engineering.

Barry Boehm is the TRW Professor of Software Engineering and Computer Science at USC, and Director of the USC Center for Software Engineering. Between 1989 and 1992, he served within the Department of Defense (DoD) as Director of the Defense Advanced Research Projects Agency (DARPA) Information Science and Technology Office and the Software and Intelligent Systems Technology Office. He also served as Director, Defense Research and Engineering (DDR&E), Software and Computer Technology Office, and as Director of two major DoD software initiatives: the DoD Software Technology Plan, and the DDR&E Software Action Plan.

- re-tooling system acquisition processes to better assess the feasibility of system acquisitions;

- digital libraries for organizing and sharing information gathered during system acquisitions and program management; and

- Internet-based electronic commerce services and capabilities for streamlining procurement actions, lead times, and supply chain logistics (Nissen, 1997; Scacchi and Noll, 1997, Scacchi, et al., 1997).

However, in this paper and in related materials (Boehm and Scacchi, 1996), we focus our discussion on the first of these areas.

Steps Toward More Agile Acquisition of Future Systems

In general terms, our overall goal is to address the recurring problems that plague system acquisition efforts. Our approach suggests ways in which new modeling and simulation techniques can help in reengineer software-intensive systems acquisition by the DoD and other government agencies. This means that we seek to identify new concepts, tools, and techniques for acquiring software-intensive systems that fulfill four goals: First, to establish viable and cost-effective system requirements. Second, to establish realistic cost, schedule, and performance estimates. Third, to mitigate against concurrent development and production of systems. Fourth, to enable adequate demonstration and testing of system viability before a commitment to system production must be made. Based on the results from a series of workshops and Blue Ribbon Panels of leading military, industry, and academic experts that addressed the problems of large-scale software system acquisition (Boehm and Scacchi, 1996), we can identify five issues involved in achieving the overall goal.

First, we need to baseline our current understanding of strengths and weaknesses of current "as-is" process capabilities for acquiring software-intensive systems. Guidelines, best practices, and lessons learned are being collected and disseminated. The Software Technology Support Center (STSC, 1995) and the Software Program Managers Network (SPMN, 1997) have assembled recent collections. Nonetheless, we also need to understand how they are employed, and to identify the operational problems that may inhibit their application and success.

"First, we need to baseline our current understanding of strengths and weaknesses of current "as-is" process capabilities for acquiring software-intensive systems."

Next, we need to develop scenarios for new "to-be" acquisition process capabilities that exploit an evolutionary "virtual" approach to the acquisition of software-intensive systems. Such an approach emphasizes the incremental acquisition of virtual prototypes for a new software-intensive system. These prototypes start as models of the intended system. These system models can be analyzed and simulated to determine which system requirements and risks have been addressed. As familiarity and confidence with the prototypes'

increases, their realism and functionality increases with the incremental integration of system components. In this way, virtual prototypes of systems can be incrementally modeled and iteratively reconfigured with simulated or actual subsystem components. The development and production of a growing number of complex electro-mechanical assemblies are now designed, tested, and refined through the use of computational models and simulations as virtual prototypes (Garcia, Gocke, and Johnson, 1994).

"The goal is to minimize cost, maximize customer satisfaction (via system performance and quality attributes) and minimize acquisition and development cycle time."

Similarly, the availability of Battle Labs suggests the use of virtual battlefields and command centers for trying out or exercising complex defense systems in alternative scenarios, through computer-based modeling and simulation test-beds operating within networked laboratories (Cothran, 1996; Wilson 1996). Accordingly, approaches such as these may also prove to be effective in supporting the acquisition of software systems. In this way, viability and cost-effectiveness of system requirements can be demonstrated, validated, and refined in an incremental manner. Similarly, estimates for the cost, schedule, and performance of an ever-more-complete actual system can also be developed and refined incrementally. Subsequently, we should also consider developing methods and scenarios for how to shift from the "as-is" to the "to-be" acquisition process we envision.

Third, we need to articulate the design and operational concept for a wide-area modeling and simulation infrastructure that's primary purpose is to serve as a test-bed and delivery platform for agile acquisition of software-intensive systems. Such an infrastructure may need to support collaboration and resource sharing between software system researchers and developers at geographically distributed sites. It may operate as a modeling and simulation collaboratory (Kouzes, Meyers, and Wulf, 1996) for software system acquisition. Similarly, such an infrastructure may need to support a hypermedia repository or digital library of technical data and information that can be accessed and shared over the Internet or World Wide Web (WWW). Such a digital library should store and organize access to software acquisitions assets. These may include publications, model and simulation libraries, reusable software subsystem components, system demonstration scenarios, multimedia presentations and annotations. In addition, the digital library may provide paths to super computing environments that support massively parallel simulations, etc.

We also want to understand how future acquisition processes or capabilities might exploit the full range of technology strategies and options at hand. The goal is to minimize cost, maximize customer satisfaction (via system performance and quality attributes) and minimize acquisition and development cycle time. Relevant technologies that can support this goal include the use of knowledge-based systems, multimedia, the Internet, electronic commerce for selling and buying software components, architecture-based software system development, high-performance

computing and communications, etc. Will new modes of academic research or industrial activity be required to most effectively support agile acquisition? If so, what are they? Similarly, what institutional or marketplace incentives are needed to help make them happen?

Finally, we need to set priorities and estimate the relative costs and benefits of candidate investments in modeling and simulation capabilities that support software system acquisition. We need to identify areas in which needs can be met largely through available technology. And we must identify areas in which acquisition research and the development of automated acquisition support environments promise an attractive return-on-investment.

Background and Foreground

We may now be at the threshold of a new era in the acquisition and development of software-intensive systems. From this point, we can look back to where we have been and what we have experienced. Then we can look forward toward the horizon to see what lies ahead.

Looking Back: Why Use Models and Simulations to Support Program Acquisitions

In looking back, we see that the acquisition and development of software-intensive systems was guided by the classic "waterfall" system life cycle. In such an approach, DoD customers were expected to be able to articulate their needs and requirements for new system capabilities prior to system development. Developers or contractors could then take these requirements as their starting point. Then they would systematically develop, test, and deliver results to the customer according to a sequence of development milestones and documentation standards.

While this approach has much rational appeal, its practice and outcome has often been less than satisfactory. The overall experience was that it was difficult for customers to fully articulate their system requirements prior to the beginning of system development. Furthermore, when system development took years, the customer (and the developer) recognized their requirements were changing, sometimes very rapidly. Consequently, far too many systems developed under contract were delivered that did not meet critical system requirements. In the worst cases, the software systems were effectively nonoperational. Subsequently, more customers and developers began to recognize that perhaps these shortfalls in software acquisition and development were systemic, rather than simply characteristic of particular programs or development organizations.

"...we need to set priorities and estimate the relative costs and benefits of candidate investments in modeling and simulation capabilities that support software system acquisition."

In response to the seemingly inevitable shortfalls with the classic approach, efforts to find an alternative began. This led to an incremental "spiral" development approach. In the classic approach, there is little visibility regarding operational software system capabilities until late in the development cycle. In contrast, the spiral

approach embraces a more evolutionary and iterative development model. Accordingly, operational software capabilities become visible in evolutionary increments, rather than all at once. Subsequent development iterations then add and integrate more increments until the final system is ready. Thus the spiral approach seeks to build and deliver software-intensive systems through evolutionary development. Consequently, guidelines now put forth in military or public standards such as MIL-STD-498, ANSI J-STD-016, and US 12207 encourage use of an incremental spiral approach when acquiring and developing software intensive systems.

"Program managers, contractors, customers, and acquisition directorate staff can use models and simulations coordinated by a negotiation support system."

Why should we use models and simulations to support the incremental acquisition of complex software systems? In simplest terms, we can identify three reasons: First, to facilitate early identification and reduction of risks associated with complex system acquisition programs. Second, to better understand what kinds of system requirements and architectures are feasible and affordable given various programmatic and technological constraints. Third, to gain insight into how to better manage the system engineering effort so as to improve the overall likelihood of a successful acquisition effort.

But the creation, use, and reliance of models and simulations to support incremental acquisition efforts cannot guarantee such outcomes. Clearly, models and simulations of complex systems will never be more than assumption-laden approximations of the systems being acquired. This is the fate of all models and simulations (Smith, 1996). Nonetheless, the process of building, using, and evolving such models and simulations in support of decision-making activities in large system acquisition efforts can be characterized as one of consensus validation (Dutton and Kraemer, 1985). Thus, the value of supporting system acquisition through modeling and simulation will be found in the process of working with them, rather than in the calculations performed along the way. Modeling and simulation can be used to help identify where consensus can be established and validated, as well as to identify where disagreements can be found, so their consequences can be examined.

Program managers, contractors, customers, and acquisition directorate staff can use models and simulations coordinated by a negotiation support system. Such a system can support the elicitation, capture, and validation of points of agreement among system acquisition participants. In addition, such a system can help these people surface assumptions, debate their merits or implications, and negotiate alternative system configurations and functional features (Boehm et al., 1995). In this manner, computer-based models and simulations, together with an information sharing and negotiation support environment, provide a more articulate medium to express opinions and stimulate alternative conceptions of system acquisition problems and challenges. Without such articulate models and simulations, system acquisition participants are left to their private intuitions and

conceptions of system design, program cost drivers, and the like. This in turn can easily obscure problems in system design or performance, increase the likelihood of miscommunication and systemic conflict, and increase the likelihood of problematic system acquisition and costly post-deployment support of the resulting systems. Thus, we believe that models, simulations, and associated environments can play a significant role in supporting the incremental acquisition of complex software systems.

Looking Ahead: An Emerging Case Study

We see many opportunities for improving the effectiveness and responsiveness of the acquisition of software-intensive systems across their life cycle. Many of these opportunities result from the availability of new technologies and development capabilities that make the acquisition of software-intensive system more agile. Agility can lead to more cost-effective, more timely, and higher quality results in software system acquisition. Modeling and simulation technologies that support virtual prototyping (Garcia, Gocke, and Johnson, 1994) and simulation-based design of complex hardware systems are being used to support major program acquisitions, such as that for the SC–21 class of battleships (SC–21, 1997). We believe a similar effort is appropriate for acquisition of the large software systems associated with such hardware systems. Accordingly, by examining the currently proposed software systems intended to support SC–21-class ships, we can better motivate and articulate a vision for how new modeling and simulation technologies can be used to help support the incremental acquisition of complex software systems.

There is no single architecture or final design envisioned for SC–21 ships. Instead, the SC–21 ships could be built following the commercial practice of developing a product line with common subsystems or reusable designs. Figure 1

Figure 1. Alternative Overall Architectures for SC–21 Ships (SC–21, 1997)

helps show what this means. Here we see four alternative views of the overall architecture of SC–21 ships. The intent to enable the choice of the final architecture of each ship to be determined by emerging need or threat. Nonetheless, any such SC–21 ship will still have some configuration of common subsystems for weapons, command deck, flight operations, etc. As such, all of the alternative versions of ship architecture displayed in Figure 1 would be members of the SC–21 product line.

Building these ships according to different architectural configurations represents a fundamental change in how such ships will be acquired, developed, and operated. The system life cycle for these ships will be iterative, incremental, and ongoing. Figure 2 conveys a vision for how various computer-based modeling and simulation technologies, such as virtual weapon system modeling and simulation-based design, may be employed to support the acquisition, development, and operation of SC–21 ships.

SC–21 ships will be software-intensive systems. All major subsystems and overall system capabilities supporting each ship's operations depend on software. Figure 3 proposes a suggested allocation of shipboard subsystem capabilities that will be implemented in software systems. Total number of software instructions or source lines of code (SLOC) to realize the proposed capabilities is estimated at

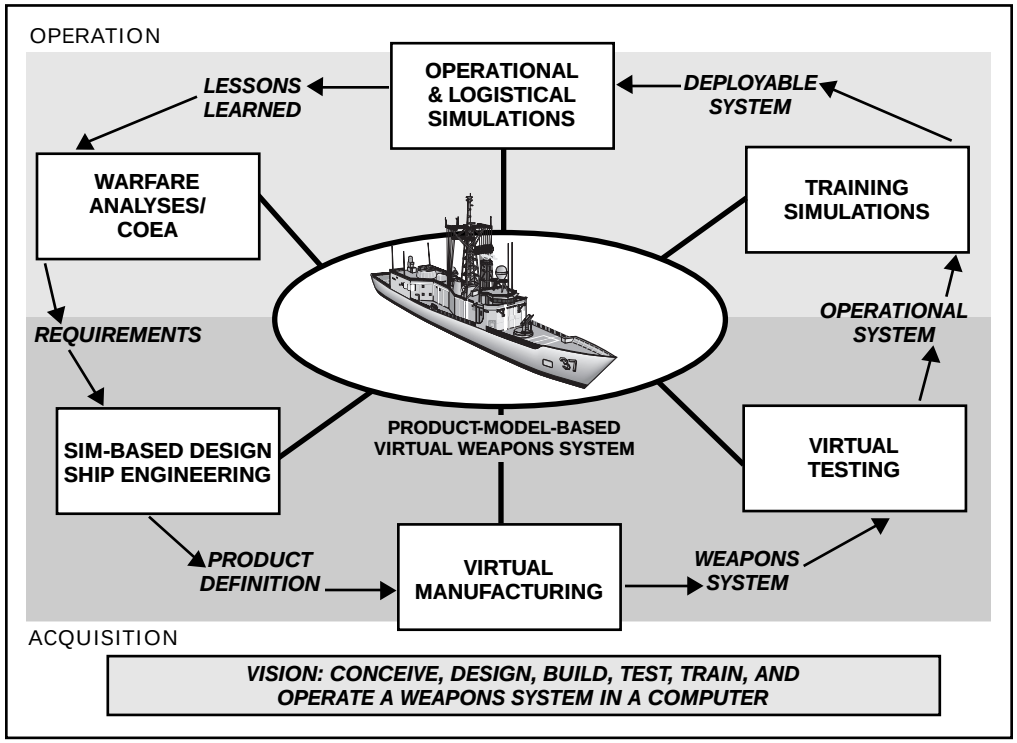


Figure 2. Vision for How Modeling and Simulation Can Support New System Acquisition and Development (SC-21, 1997)

greater than 8.4 million SLOC. Much of this software can potentially be reused across the SC–21 line of ships, however. Nonetheless, development costs for software of this size and complexity is often estimated in the range of $100 million to $1 billion. Thus, what can be done to help understand the feasibility of alternative software subsystem architectures associated with the SC–21 ship family, and manage the progress, costs, and risks associated with the acquisition and development of this software?

At present, there is an emerging consensus for what technological capabilities are needed to support the acquisition and development of software-intensive systems such as the family of SC–21 ships (Boehm and Scacchi, 1996). Much like the SC–21 family of ship hardware and major subsystems employs recent advances in modeling and simulation technologies, similar technologies could be brought together to support the acquisition and development of the software systems for these ships. Accordingly, we can now outline a strategy for how this would work. We then follow with a discussion of the technological and organizational transitions likely to be encountered in the course of adopting this strategy. Along the way, we describe an approach for how to assess the feasibility of complex software systems through its incremental development spiral. In addition, we describe a road map that lays out the research, technol-

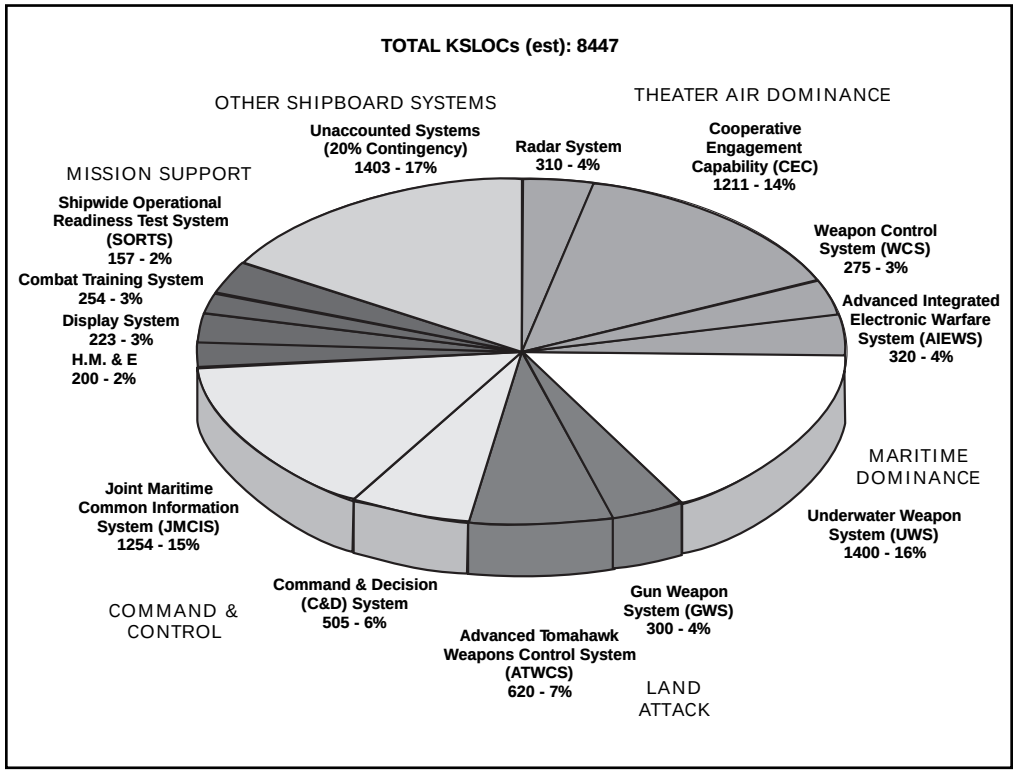


Figure 3. Software Systems Proposed for SC–21 Ships (SC–21, 1997)

ogy, and usage needed to support the acquisition of software systems, such as those for the SC–21 line of ships.

The Virtual System Acquisition Vision

The virtual system acquisition (VISTA) of software systems refers to a strategic process by which an evolving series of ever more complete and operational system versions are acquired through a series of short duration acquisition life cycles. In this way, emphasis is on reframing and reducing acquisition cycle times from years to months (or weeks!) so as to focus attention on the incremental and iterative acquisition of the evolving capability associated with the target software system.

"The virtual system acquisition (VISTA) of software systems refers to a strategic process by which an evolving series of ever more complete and operational system versions are acquired through a series of short duration acquisition life cycles."

Reductions in acquisition cycle time enable an increase in the number of incremental acquisition cycles over time. The VISTA approach seeks to help more rapidly identify, address, and resolve the risks associated with the acquisition and development of complex software-intensive systems (Boehm and Scacchi, 1996; GAO, 1997; Haimes, Schooff, and Chittister, 1997). Thus, we need tools that enable customers and developers to rapidly model, incrementally evolve, and satisfy (sub) sets of system capability requirements in each iterative system version. Early acquisition cycles need only to focus on acquiring systems that represent computational models and simulations of the operational capability of the target software system. Later acquisition cycles then focus on incrementally evolving or replacing the models and simulations with fully operational system modules. In this manner, there will always be an operational version of the system to evaluate and demonstrate throughout the system's acquisition and development cycle.

Models and simulations represent descriptive, formalized, and sharable understandings of a system. They can represent a system's concept of operation, architecture, and its ability to support its intended mission. However, by focusing effort to enable such preliminary system capabilities to move through a fast acquisition life cycle, the goal is to establish and validate consensus on whether current models and simulations of the software system's components or architecture address specific system requirements. In addition, the goal is to determine whether other underdeveloped or unrecognized system requirements have emerged that must be addressed in subsequent acquisition and development cycles. As such, the goal here is more closely aligned with the idea of incrementally growing and evolving the target system in a more organic and adaptive manner.

Our first take on the requirements for how this might work can be outlined as follows:

- Acquisition participants should be able to architect, construct, assemble, execute, and analyze automated models of the overall software system capability for acquisition.

- Component models should represent elements of target environment (including people), information system infrastructure, informational products, and development, operation, and post-deployment processes.

- The initial modeling and simulation of these elements should represent the first pass through the system's requirements generation and development cycle.

- Participants should be able to iteratively refine and incrementally evolve the system model test-bed from previous steps. Then they should be able to selectively replace component models with simulated, prototype, or actual component elements.

- Participants should be able to iteratively refine and evolve intermediate hybrid system test-beds and progressively replace remaining component models with simulations, prototypes, or actual component elements. This helps to insure that a full-scale test-bed is developed, operational, and ready for post research and development deployment or transition into commercial use.

Subsequently, we can take this outline of requirements for what we envision and reformulate it into a first-cut prescriptive process, which we call the VISTA Approach.

The VISTA Approach

At this point, we outline a series of steps that articulate how software system acquisition and development become intertwined during virtual system acquisition processes. As modeling and simulation drive most of these steps, we first describe what types of models are necessary. We will also characterize what these models may look like, and how they could be represented. Then we will briefly describe how these models and simulations would be incrementally replaced when evolving the system.

Modeling and Simulation in VISTA

For this discussion, we assume the envisioned system is within the scope of available software system product families at hand. If not, then a domain analysis leading to the construction and refinement of an appropriate meta-model will be needed. Product families and their associated "smart" product models (SC-21, 1997), documents, development processes, tools, and organizational agents are defined and represented using meta-models. Detailed examples of their use can be found elsewhere (Mi and Scacchi, 1990, Mi and Scacchi, 1996, Scacchi and Mi, 1997). We then begin with the elicitation and modeling of a virtual system model (VSM) for the system to be acquired.

The VSM is a composite model—a model composed from other models. At least three types of models are needed to characterize a complex software system. One class of models is needed to represent the functional operation and data required for information processing by the system. We will call models of this type information element models (IEMs). Once an IEM is replaced with an operational system component, it becomes an information element (IE). IEMs are used to model the structure, behavior,

and performance (estimated, measured, or required) of the computing hardware and software that inputs, processes, and outputs system data. A second class of models is needed to depict the functional behavior of the IEs embedded within a man-machine system (e.g., command and control system, theater air dominance system, mission support system, etc. in Figure 3) to be acquired and built. We call these system element models (SEMs), and when replaced, system elements (SEs). The third class is needed to represent the "system of systems," sensors, and environmental context in which the embedded man-machine systems operate. These are called environment element models (EEMs), and when replaced, environment elements (EEs).

"Each type of model requires a computational mechanism that can support model entry and definition, interpretation, simulation, and animated visualization."

Each type of model requires a computational mechanism that can support model entry and definition, interpretation, simulation, and animated visualization. Commercially available discrete-event simulation packages represent one such mechanism. These packages are well suited for simulating models that are represented as queuing networks whose arrival queues and service rates are specified according to statistical or algebraic models.

Different types of models may require different kinds of simulation; thus different tools may be needed. For example, modeling and simulating the "look and feel" and event-based operation of a graphic user interface for a Military Support Training System may employ multimedia authoring or navigation tools. Commercially available tools such as Macromedia Director, Microsoft Powerpoint, or even Web browsers accessing virtual reality content across an intranet can be used for this purpose. Rapid application development (RAD) tools (Visual Basic, PowerBuilder, Visual Cafe for Java, etc.) and expert system shells (e.g., M.4 from Teknowledge) that support software prototyping or visual programming with persistent databases can enable the modeling and simulation of complex, rule-based, state-transition software applications. These are tools for developing virtual prototypes of IEs (Garcia, Gocke, and Johnson, 1994). With these tools, it is possible to model, simulate, or approximate the behavior of software applications using stubbed, canned, or pre-calculated input and output data values as place holders for complex calculations required of an eventual software system implementation. As such, modeling and simulating a VSM may benefit from use of a computing environment where multiple types of models and simulations can be defined, composed, simulated, and displayed. Furthermore, it may be desirable for such an environment to be accessible over the Internet to facilitate the sharing, discussion, and review of modeling and simulation efforts among the different organizational representatives participating in a program acquisition.

IEMs can be modeled in a variety of ways. A common tactic may be to depict IEMs as hierarchically decomposed black boxes (closed systems), white boxes (open systems), or gray boxes (closed systems with limited internal visibility). These boxes are placeholders for hardware or

software system modules that are to be acquired and developed. Each box can represent a computation unit that can receive inputs or event signals, perform some calculation, then produce some outputs, state transition, or some new event. They can be modeled and simulated using any of the tools noted above. However, depending on the kind of acquisition concern we wish to address, particular tool choices may be most appropriate. For example, in SC–21-class ships, it may initially be an open question as to what level of computer performance is required to satisfactorily operate mission support software systems. A desktop personal computer is probably inadequate, while a large mainframe may be too much, too large, or too expensive. Thus, it seems appropriate to consider modeling the required computing hardware as a computational module with mid-range performance or processing throughput (i.e., 10–100 transactions per second) as a starting point. Further, since determining system performance throughput under different mission support workloads or traffic volume is necessary, then a discrete-event simulation package may be best to use.

However, the software system modules required to operate on this anticipated hardware may or may not be so readily understood. If we initially have little knowledge of what calculations or information is required in processing mission support data, then the software's model may simply equate to that of a module that produces a stream of input and output data transactions, say in the range of 0–8 transactions per second. Alternatively, as knowledge increases, software modules may be identified that perform different functions.

It should be possible to evaluate alternative architectural configurations or compositions of software modules as a way to understand whether system performance parameters are sensitive to the alternatives.

For example, in a mission support combat training system, one could separate user input capture and verification, calculation and database update, and output to user display as three distinct software modules. Should these modules be configured in as a linear sequence, a fully interconnected concurrent network, or bundled together as a single large module? Which alternative configuration would be easiest to build and test? Which would have the best performance? Which would cost the least? Perhaps we could guess the answer(s). However, if we can model, simulate, and collaboratively discuss the three architectural alternatives, then we can begin to articulate a basis that leads to a consensus answer that can be backed up with evaluated alternatives and simulation results.

"It should be possible to evaluate alternative architectural configurations or compositions of software modules as a way to understand whether system performance parameters are sensitive to the alternatives."

Would the consensus results from such a modeling and simulation exercise be more believable than someone's best guess? In lieu of some controlled experiment, the answer to that is subjective. However, the modeling and simulation results would be explicit, repeatable, and subject to tradeoff analysis and consensus validation. In addition, these results can be open to challenge and reformation

in a manner that may be more tractable than someone's best guess. Nonetheless, if someone such as a software architect experienced in the design of mission support combat training systems can argue persuasively about his or her best guess, then this alternative could be represented in an IEM, simulated, compared and validated.

SEMs provide the ability to embed software systems within man-machine systems settings. SEMs embed IEMs or IEs in a user-driven input and output environment. Users create inputs in response to their work assignments, and to information output from the system and displayed to them. For example, when using a training system, users may select among "menu items" or enter system commands. This may cause the training system to process their input, provide an updated user interface display, then wait for the user's next input action. As such, SEMs must model user behavior in driving and responding to system actions or events, as well as model system behavior in response to user actions.

"...users can only provide either acceptable input, acceptable but erroneous input that is detected, or unacceptable input."

While user behavior is open-ended, only a range of possible user-system interactions will be modeled. For example, users can only provide either acceptable input, acceptable but erroneous input that is detected, or unacceptable input. SEM simulation may include the use of software "drivers" that cause the arrival of user input or input events, together with system responses or service time intervals that follow statistical formulas or some other characterization function. SEM simulation can then be supported using common discrete-event simulation tools if user behavior is being simulated. Alternatively, if the system's behavior is being simulated for real users, then multimedia or RAD tools may be used to provide simulated user interfaces for real users to evaluate. As with the IEM simulations, the plausibility and consensus validation process noted above will also apply here.

EEMs provide the ability to embed the man-machine system in its overall environmental context. For example, weapons control systems may be designed to use various sensors (radar, sonar, satellites, etc.) to zero in on their targets. These sensors may themselves be complex systems. Similarly, weapons control systems will interact with many other shipboard systems, including those for mission support, and command and control. These systems must act in concert to realize the overall effectiveness of a complex system of systems that a ship of the SC–21 class represents. Therefore, EEMs must model the interoperation and integration of multiple systems. This may entail modeling the overall patterns of data or messaging traffic between systems, as well as between systems and users as a group.

Alternatively, in response to different scenarios for total system engagement, the EEMs may be used to model the ebb and flow of information across the system of systems. With this, we expect that the patterns of information flow on a SC–21 ship in response to a hostile attack scenario will be different than the flows associated with routine ship operations and maintenance scenario. Subsequently, these information traffic or flow patterns can be modeled

and simulated using discrete-event simulation capabilities, and the validation process described earlier again applies here.

Overall, the remaining challenge is to integrate and interoperate the different models, simulations, and elements. This is the purpose of a collaborative test-bed such as a Battleship Lab for SC–21 class ships (Cothran, 1996; Kouzes, Meyers, and Wulf, 1996; Wilson, 1996). It may serve to support the integration and interopera-tion of multiple, mixed mode models and simulation tools, as well as of multiple system elements with many models and simulations. At this time, developing such a test-bed may be an expensive but nonetheless necessary proposition. However, even if the cost of such test-beds approaches 5-10 percent of system development costs, such an investment may be reasonable given that the total overall effectiveness of the system platform is long-lived, software-intensive, and thus software-dependent.

Again, our objective is to find ways to facilitate the articulation and elaboration of requirements, risks, and cost-drivers for complex, software-intensive systems. It also assists those involved in system acquisition to understand how modeling and simulation tools and techniques can be used. As such, we now provide a brief description of how incremental system acquisition and development would replace the system models with operational elements and system components.

Incremental Replacement of System Models with Operational System Components

Given that we have outlined the overall VISTA approach for modeling and simulation, we can describe how this approach could work in the context of acquiring a software system. We examine software systems for SC–21 class ships, although we limit our discussion to a representative subset of software systems for these ships. We use mission support systems in our discussion. Accordingly, we describe how the information, system, and environment elements for mission support are incrementally acquired and developed in a series of spiraling iterations following the approach. We show how these elements can change while progressing from models to actual software system architectures. Similarly, we identify what difference it makes to improve the acquisition of software.

"Again, our objective is to find ways to facilitate the articulation and elaboration of requirements, risks, and cost-drivers for complex, software-intensive systems."

The VISTA approach begins with the acquisition of a virtual system model for mission support. A team of participants from the program office, acquisition directorate, user representatives, and prospective contractors may specify the VSM. The team might employ a wide-area collaboratory environment to share and record information giving rise to the VSM. However, perhaps only the contractors would be tasked with the modeling development activity.

The VSM can be subjected to analysis, simulation, redesign, visualization, and walk-through. Figure 4 provides a concept diagram for how this might appear if we focus on an architectural configuration of IEMs (the computer or software

elements), SEMs (the physical or human elements), and the EEMs (the external stimuli outside the system boundary). As shown in Figures 4 through 6, multiple IEMs, SEMs, and EEMs are used. This reflects the notion that the scope and depth of different models may be limited, compartmentalized, or may be divided among different organization contractors, sub-contractors, program office, etc.).

In acquiring an initial VSM for mission support systems, many kinds of models are used. For example, IEMs designate computer hardware and software subsystems. SEMs denote operational readiness test system, combat training system, and display system. Also, EEMs are needed for other shipboard systems (e.g., command and control system), sensors, and environment factors (weather, combat versus routine operations, etc.). Emphasis in developing the initial VSM is on deciding what kinds of modeling and simulation tools to use for the different types of model elements. Also, emphasis

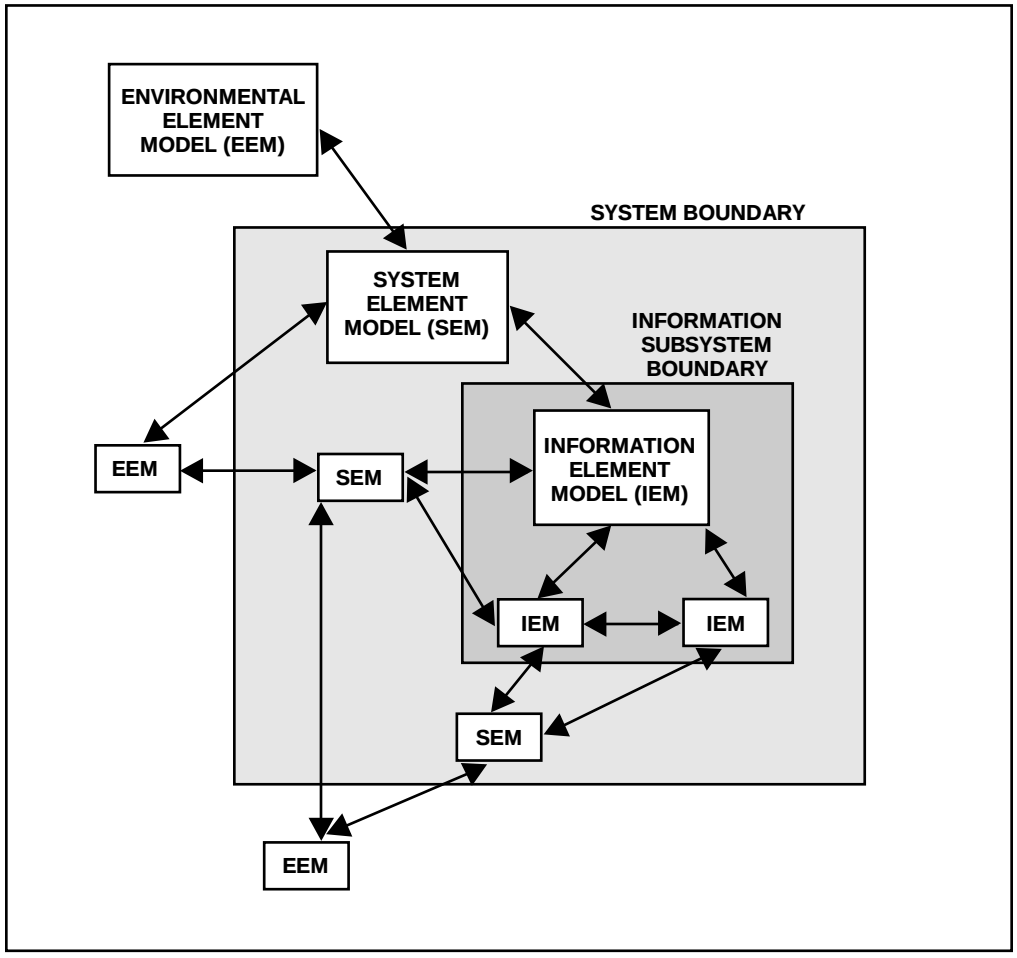


Figure 4. Initial VSM Development Cycle

is directed at how to integrate the modeled elements into an architectural configuration so that the simulated elements can interoperate. This is shown in Figure 4. Subsequently, if all VSM element models can be satisfactorily simulated at this point using a discrete-event simulation package, then the integration and interoperation challenges are reduced or eliminated.

Given that the VSM can be developed, we need to exercise and test it to explore the proposed system's ability to satisfy the requirements of its customers, users, development contractors, program managers, and others. Similarly, we need to explore the tradeoff among desired system functional capabilities, performance objectives, and costs. A wide-area software requirement negotiation and collaboration environment, such as the Win-Win environment developed at USC (Boehm et al., 1995), could be used for this purpose.

Collaboration environments, like Win-Win, enable various system acquisition and development participants to discuss the relative merits of the VSM, its ability to identify or demonstrate system requirements, and to determine and validate where there is consensus in these areas. For example, user representatives may believe that response time to user input commands should not be more than one second. The contractors may note that while such system performance may be essential for the combat training system, it may not be needed by the operational readiness test system. Thus, it would be unnecessarily costly to the program to make it so.

To help clarify their position, the contractors input the two alternative system performance requirements into the computer hardware IEM simulation. Executing the simulation using the two performance measures may produce interesting comparative results. For instance, if users of the operational readiness test system can accept a four-second response time, the required computer hardware performance can be realized at an appreciably lower cost, perhaps saving millions of dollars (Boehm and Scacchi, 1996). With this result at hand, the team agrees to revise the requirements for this information element. As such, the VSM is revised and calibrated to use this information. This helps to illustrate the how iterative analysis, simulation, performance monitoring, and benchmarking can improve understanding system requirements, and how to identify areas where virtual system acquisition efforts can reduce costs.

"...we need to explore the tradeoff among desired system functional capabilities, performance objectives, and costs."

In a later acquisition and development cycle, the team decides to assemble particular element components using fully operational and architecturally configured subassemblies. Here, the contractors must replace the corresponding model or simulation elements with operational prototypes or actual operating elements. Figure 5 provides a diagram for how this hybrid system and hybrid test-bed might appear. For example, an EEM for sonar and radar sensors may be replaced with a test-bed instrument that can generate realistic sensor input data. The display system for mission support may now be fully operational, and the computer hardware that supports the display system may be operational. Accordingly, the display

system SEM can be replaced with the operational display system SE, and the computer hardware IEM can be replace with its corresponding IE. Nonetheless, even with these virtual system elements replaced with operational components, the overall VSM test-bed can still be accessed and evaluated using a collaborative wide-area environment for requirements negotiation and validation (Boehm et al., 1995, Kouzes, Meyers, and Wulf, 1996).

Once operational components are integrated into the VSM, it becomes possible to more systematically walk through, exercise, monitor, record, and replay the revised VSM hybrid tested. This can help to validate choices, explore further tradeoffs, and articulate systemic bottlenecks or processing failures in the system's architecture (Scacchi and Mi, 1997). For example, while evaluating the operational performance of the display system that interacts with the combat training system, it appears to users that important information of the user display is being updated too fast for users to act

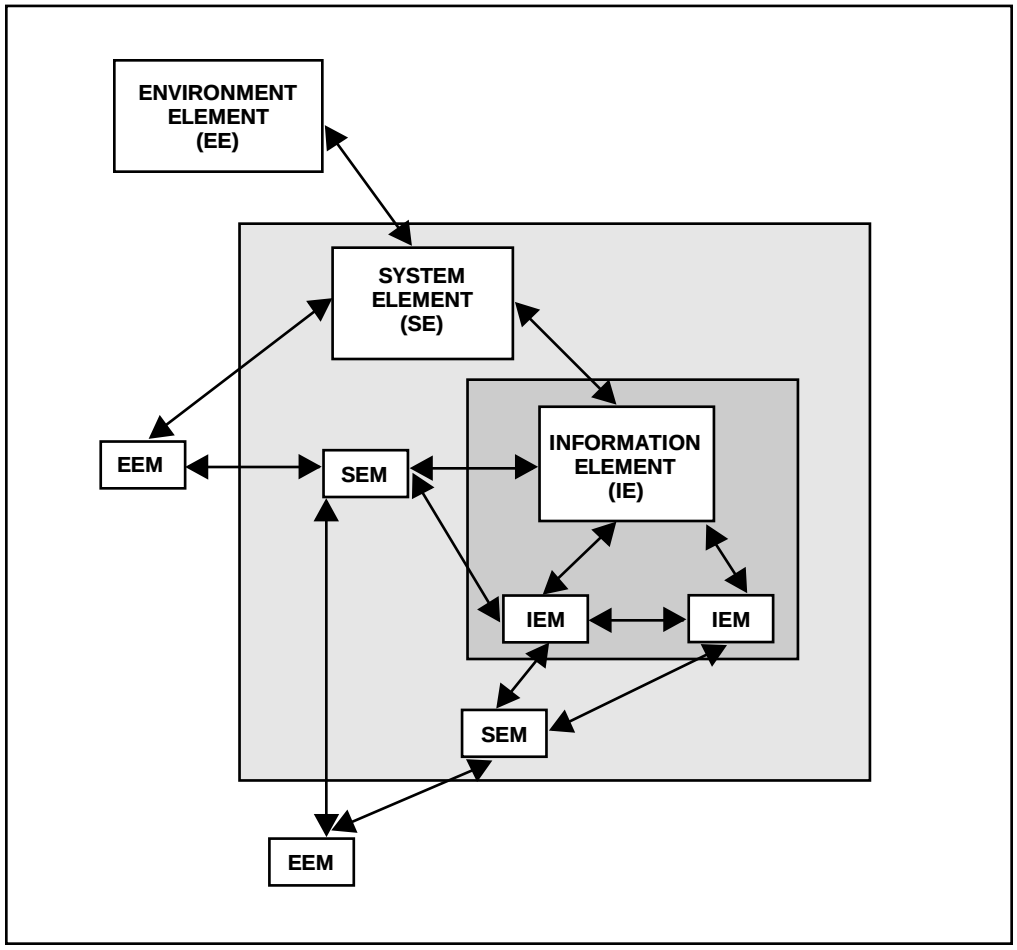


Figure 5. Intermediate VSM Development Cycle

appropriately. Instead, the rate of information display needs to be slowed, or the information content needs to be aggregated and summarized. Thus, from the user standpoint, the current system operation in the VSM is risky or not feasible. As before, system element parameters need to be adjusted, otherwise alternative system architectures need to be considered and evaluated.

With an intermediate VSM, further elaboration is needed to field a deployable system (see Figure 2). If this is the case, then the acquisition and development team must revisit the selection of software and system components to develop. Otherwise, they can perform partly simulated operational test and evaluation, then experimentally field the system either across a wide-area intranet test-bed (Scacchi and Noll, 1997), or in a battleship lab test-bed, in order to continue to calibrate and refine the VSM for further post deployment studies. Thus, here we seek to illustrate how virtual system acquisition can help identify potential risks and attendant cost drivers that may not be manifest until field operation stages of the system's overall life cycle.

When further system capabilities are needed, the participants can exercise the VSM. This means they may adjust simulation parameters, have users test-drive and evaluate system prototypes, etc., to determine tradeoffs and validate priorities through consensus. Consequently, they may choose to revisit the selection of components to acquire and develop. Jumping ahead, the acquisition and development participants can continue to evolve and continuously improve the emerging system architecture. This requires a revisit through the preceding steps until all remaining system component simulations or prototypes are replaced by their operational counterparts. Figure 6 provides a diagram for how this late stage system architecture might now appear.

Here we see that all of the system and information element models have been replaced with their operational elements. Some EEMs remain, however, since they may designate other major shipboard system undergoing concurrent development. Thus, while the sensor test-bed may be operational and integrated to interoperate with the mission support systems, the command and control system as well as other major systems may not yet be operational and available for integration. But these other systems must still conform to their EEMs placeholders for use with the mission support system. Subsequently, an additional capability is required for characterizing or extracting an updated EEM from this VSM. This updated information needs to be used in other VSMs corresponding to environment elements that constitute the system of systems. From a technical standpoint, this requires addressing problems in system component interface definition, and in managing concurrent access to different versions of these components or model placeholders. From an organizational standpoint, failing to coordinate access and propagation of component interface definitions or changes is a common problem that precipitates

"Once operational components are integrated into the VSM, it becomes possible to more systematically walk through, exercise, monitor, record, and replay the revised VSM hybrid tested."

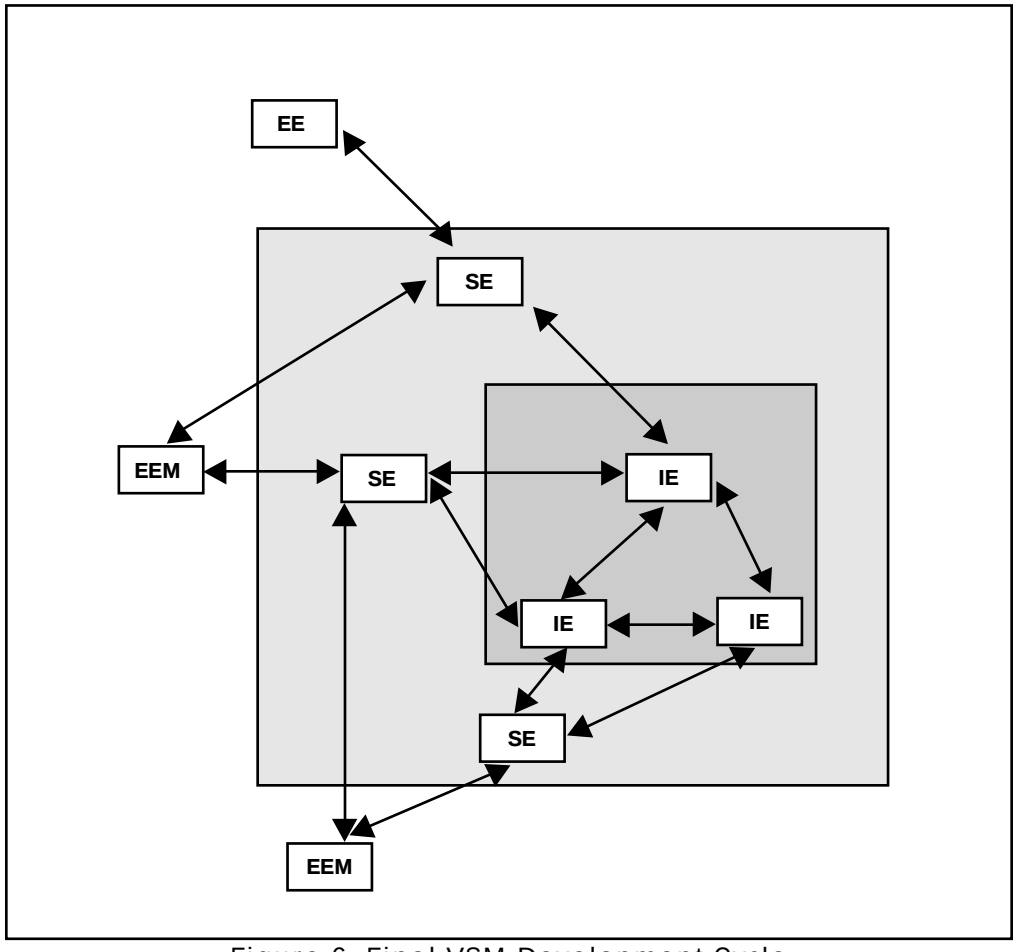


Figure 6. Final VSM Development Cycle

difficulty in systems integration and interoperability. Knowing where problems lie, and being able to prevent or circumvent them through virtual system acquisition, provides another capability for reducing risks and costs associated with the development of software-intensive systems.

Finally, throughout the overall VISTA process we have just outlined, current best practices in software program management (SPMN, 1997) and a consensus recommendation from the Blue Ribbon Panels (Boehm and Scacchi, 1996) point to the opportunity to track and manage software feasibility and risk using new program management support tools. Figure 7 provides a view of the user interface "dashboard" to such a tool, as well as suggesting how program management information may be conveyed.

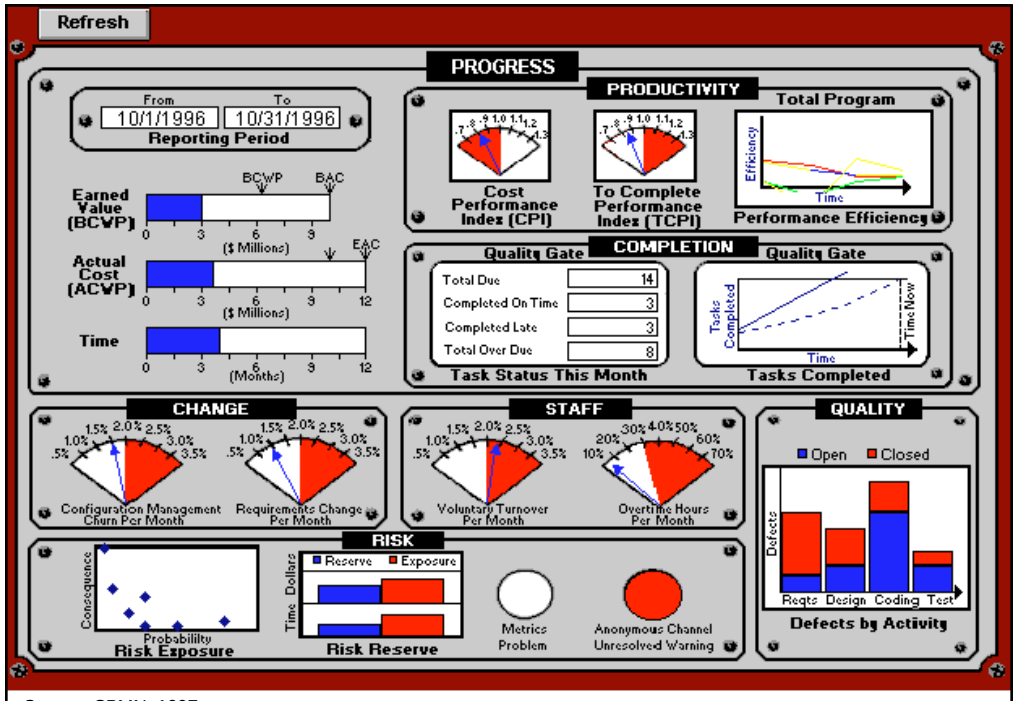
Participants in a virtual system acquisition also need to track, organize, record, and store records of the steps they took. Furthermore, they may need to document what transpired, how, by whom, why, and with what outcomes. These records and

documents represent important knowledge assets emerging from the acquisition effort. Capturing and organizing this information is often cumbersome and haphazard. However, we find that these knowledge assets can be easily captured and linked to the virtual system models and elements using hypertext mechanisms commonly available in information sharing and requirement negotiation support environments (Noll and Scacchi, 1991, Boehm, et al., 1995), rather than being cast as a mountain of paper.

With this basis for VISTA approach, we can now put forward a matrix of the transitional steps for how to realize the technical basis for supporting VISTA. This is then followed by a description of the organizational transitions for VISTA.

Mapping the Technological Transitions to VISTA

Although the VISTA-based approach may be a radical departure from traditional system acquisition practice, getting there may be best achieved in an evolutionary manner. To be clear, the VISTA approach is new, but the tools, techniques, and concepts it involves—incremental acquisition and development, virtual prototyping, wide-area collaboratories, software requirements negotiation and validation environments, etc.—are beginning to be used in system acquisition efforts. Thus, as VISTA implies the need to use an automated support environment for modeling, simulation, and program management, the



Source: SPMN, 1997

Figure 7. A Program Management Dashboard for Assessing Software Development Progress

required tools and techniques for such an environment can be investigated, refined, and deployed in a multistaged manner. An integrated information management environment to support the acquisition and development of complex software systems, such as those for the SC–21 program, is not yet available. However, such an environment can be constructed and put into use following the road map outlined below and elsewhere (Boehm and Scacchi, 1996). The resulting environment can then be positioned to support large system acquisition programs.

We can explain the technological basis to support the transition to VISTA in terms that cover its anticipated use in acquisition, (its technology, and the research needed to realize its technology and usage. At the same time, we can characterize how each of these three aspects correspond to the software system development life cycle stages that include system concept definition, architecture definition, and ongoing spiral development. Together, we can associate each of these into a matrix that organizes the VISTA research, technology, and acquisition usage as shown in Table 1.

"To be clear, the VISTA approach is new, but the tools, techniques, and concepts it involves... are beginning to be used in system acquisition efforts."

Moving from top to bottom, right to left, we can outline the associated operational concepts for VISTA, thereby characterizing the technological transitions "from ends to means."

- Concept feasibility determination: Given a new mission or strategic objective, determine whether appropriate technology, architectures, and resources can be feasibly brought together into a new software-intensive system in an affordable and timely manner.

- Architecture feasibility determination: Given a proposed software system architecture, determine whether it can satisfy mission or strategic objectives in an affordable and timely manner.

- Virtual system acquisition: Given a feasible system concept and architecture, acquire the proposed architecture as a series of modeled, simulated, or implemented subsystems. These subsystems can be developed by progressively replacing or transforming the modeled or simulated subsystems with prototyped or real implementations.

- VISTA-1, top-level feasibility advisor, parametric models: A top-level feasibility analysis-modeling environment is needed for checking established acquisition heuristics and parameters. Such an environment could be used to determine whether the candidate technologies, architectures, and resources can be brought together to address a new mission or strategic objectives. This environment would represent the first version of the VISTA support environment (VISTA-1). The environment proposed by the software program managers network (Figure 7), together with software cost estimation tools, software requirements negotiation capabilities, and access to a collection of software feasibility heuristics are available today for

Table 1. VISTA Research, Technology, and Usage Context

		TECHNOLOGY MATURITY		
		Research	**Technology Usage**	**Acquisition**
SOFTWARE/SYSTEM LIFE-CYCLE STAGES	**Concept Definition**	Software feasibility heuristics	VISTA-1: Top-level feasibility advisor, parametric models	Concept feasibility determination
	Architecture Definition	Architecture representation and analysis M&S, advanced cost/schedule/ quality M&S	VISTA-2: Models and simulations of subsystems and elements	Architecture feasibility determination
	Spiral Development	Integration into commercial SDEs	VISTA-3: Hybrid measurement, M&S environment	Virtual system acquisition

experimentation and initial usage (Boehm et al., 1995; STSC, 1995; SPMN, 1997).

- VISTA-2, software-intensive models and simulations: VISTA-2 is an enhanced VISTA-1 environment for software-intensive modeling and simulation. It could be used to prototype, analyze, and execute system architectural capabilities and functionality, then reconcile these performance characteristics against the cost, schedule, and quality tradeoff among proposed architectural design alternatives. VISTA-2 is used order to determine whether proposed application system architectures are viable.

- VISTA-3, hybrid measurement, modeling and simulation environment: The VISTA-3 environment is built to expand the capabilities of VISTA-2. In order to acquire incrementally developed software application systems, VISTA-3 can be used to support the cooperative modeling, simulation, and measurement of the performance capabilities of an evolving application system, its subsystems, and their collective architectural design.

- Software feasibility heuristics: We need to collect, validate, and refine a knowledge base of "best practice" heuristics for software system acquisition, architecture, and overall development.

This knowledge could provide plausible advice for how to assess the top-level feasibility of an emerging software application system. These heuristics can help determine what matters, and which technology, architecture, or resource characteristics affect the overall feasibility of the system (Rechtin, 1991; STSC, 1995; SPMN, 1997).

- Architecture representation and analysis modeling and simulation (M&S), and advanced cost, schedule, and quality M&S: We need to research and develop new architectural representations that support incremental building and evolving large application systems using models or simulations. These representations also must be able to incorporate the architectures of its subsystems, whether as already implemented or newly developed components. We further need to be able to represent the cost, schedule, and quality associated with the development of different software components or architectural configurations.

"We need to collect, validate, and refine a knowledge base of best practice heuristics for software system acquisition, architecture, and overall development."

- Integration into commercial software development environments (SDEs): In order for VISTA tools to be broadly applied across the spectrum of DoD or other large-scale system acquisitions, they need to become available as extensions (e.g., "plug-ins" or "helper applications") to commercially available software engineering environments.

With this context for VISTA research, technology, and acquisition usage in mind, we can now more simply characterize the overall concept for how VISTA might be employed. This can be outlined in four steps:

- Pre-proposal requirements analysis: Use the VISTA environment to analyze feasibility of the system's concept and mission requirements (a sanity check on the technical perspective for a new mission program to determine rough order of magnitude for cost, architecture, other risk items, etc.) prior to the Request For Proposal.

- Proposal analysis: Upon receipt of development contractors' proposals, use VISTA to analyze each proposal for feasibility, determine which proposals are in competitive range, and what assistance is needed to evaluate the technical perspective (e.g., architecture) of those proposals within competitive range.

- Project startup: Use VISTA to evaluate the feasibility of resources (cost, people, etc.) and schedule of proposed system design. This could also be used for "fly-off" scenarios as well, when competing designs are being evaluated.

- Ongoing program review: Use VISTA to re-analyze feasibility at progress milestones during development life cycle, as well as when significant program or system requirements changes occur.

VISTA should be applicable to product-line software system architectures, as well as to unique non-product-line software systems. It appears that the VISTA may be more readily suited to product-line software system architectures, since their recurring development can accommodate the collection, refinement, and calibration of the VISTA for the product line's application domain. However, it may also be useful for (portions of) non-product-line software, especially where a well-conceived reference model standard, such as the Air Force's Horizon Architecture, defines the software. Nonetheless, within the domains of C4I, air traffic control, management information systems, and other applications, we may expect future systems to be more likely to conform to product-line architectures. Industry trends and corporate strategies may then lead system development contractors to focus their expertise and core competencies around the mastery of product lines, rather than individual products or contracts.

Managing the Organizational Transitions to VISTA

The move to adopt, implement, make routine, and replicate the VISTA approach seems to be a radical departure from current system acquisition practices and processes. While we believe that a compelling technical argument can be made for the VISTA approach, we must also address the kinds of organizational situations or changes that must be part of the transition to VISTA.

Personnel will be unfamiliar with VISTA and what is required to reengineer the processes they enact during system acquisition. Mutually respected collaborative education, elicitation, and information sharing among the participating user, development contractor, and program management organizations will be required. WWW-based collaborative work environments or acquisition collaboratories (Kouzes, Meyers, and Wulf, 1996) can help provide the information infrastructure needed to support this. But participation and engagement in reengineering system acquisition, development, and program management must span all levels of the organization chart, and must achieve commitment, resources, and strategic attention from executive and senior management in order to increase the likelihood of success (Bashein, Markus, and Riley, 1994).

Our characterization of "as-is" system acquisition processes and practices, as well as "to-be" VISTA based approaches are understated. Clearly, there is far more detail to system acquisition or virtual system acquisition processes and practices than can be described here. Furthermore, we recognize that both "as-is" and "to-be" approaches to system acquisition are put into practice in different ways, in different organizational settings, for different system acquisitions. Capturing, understanding, and describing these variations requires systematic research, empirical investigation, and wide-area dissemination. However, experience has shown that this attention to detail can lead to

"The move to adopt, implement, make routine, and replicate the VISTA approach seems to be a radical departure from current system acquisition practices and processes."

distinguishing what's common from what's circumstantial. Such detail will help surface specific actions to take to successfully engage personnel to collaborative identify and perform the organizational transformations needed to transition from the "as-is" to the "to-be."

Next, as the world moves towards a globally networked information infrastructure based on the Internet and WWW, we recognize that the information systems and computer-based tools supporting the acquisition, development, and program management will increasingly become heterogeneous relative to one another (Noll and Scacchi, 1991; Scacchi and Noll, 1997). Interoperability will not be easily achieved without the experience and expertise needed to make it happen. However, new information technologies are rapidly emerging that will give rise to new ways to more rapidly configure, interconnect, and integrate software systems in order to enable them to interoperate. Furthermore, what's likely to be critical during early VISTA-based acquisition and development cycles is realizing interoperability at the organizational process level, rather than only at the traditional system function level. Experience shows that addressing and resolving interoperability between distinct organizations, such as those participating in a system acquisition, can often lead to ways to obviate, minimize, or avoid system function interoperability dependencies (STSC, 1995). This helps to refine, streamline, and focus both system architecture and system development processes.

"Next, as the world moves towards a globally networked information infrastructure based on the Internet and WWW, we recognize that the information systems and computer-based tools supporting the acquisition, development, and program management will increasingly become heterogeneous relative to one another."

Last, as indicated earlier, attention in this article is directed at emphasizing the re-tooling and reengineering system acquisition processes and system feasibility assessment. However, a greater payoff can potentially result from complementary incorporation of process reengineering concepts, techniques, and tools into VISTA approaches (Nissen, 1997; Scacchi and Mi, 1997; Scacchi and Noll, 1997; Scacchi, et al., 1997). For example, recent efforts to redesign acquisition and procurement processes for the Navy have identified a number of ways these processes can be transformed and streamlined to realize substantial reduction in cycle times and administrative costs (Nissen, 1997; Scacchi, et al., 1997). But these capabilities have not been used to support the acquisition of large software systems and thus require further investigation. Nonetheless, the vision of a 21st century "digital government" raises such matters for systematic acquisition research and empirical investigation befitting a grand challenge to the academic, industrial, and government research community (Schorr and Stolfo, 1997). Subsequently, the acquisition community needs to stimulate research that can find new ways to radically streamline program operations, reduce system costs, and improve service quality through reengineering, reinvention, and systematic utilization of emerging information technologies and infrastructures.

CONCLUSIONS

We have identified opportunities for research and application of modeling, simulation, and evolutionary development technologies to re-tooling and reengineering system acquisition processes. These tools and techniques can help to analyze overall feasibility and risks at various points in the system acquisition life cycle. Such a capability offers the potential to reduce software system acquisition risks and avoidable costs, as well as explore alternative system options in order to develop more affordable, capable, and flexible systems. Subsequently, we use the new SC–21 battleship program as a case study to help illustrate and explain how virtual system acquisition can work.

We put forward a vision and approach for how to rethink the manner in which software-intensive systems can be acquired across the acquisition life cycle. Central to this vision is a new approach to virtual system acquisition we call VISTA. We believe that VISTA offers a new strategy for how to address, resolve, or mitigate the recurring problems that accompanies complex system acquisition. Major program acquisitions such as the SC–21 class of ships, the Joint Strike Fighter, and others are positioned to take advantage of timely investment and adoption of VISTA strategies and support environments.

VISTA is a new approach to the acquisition of software-intensive systems. It seeks to build on knowledge of best practices in "as-is" acquisition and development processes, as well as moving toward a re-tooled and reengineered "to-be" software systems acquisition and development process. The acquisition of complex systems such as the SC–21 class of ships will use virtual prototyping and manufacturing tools to acquire and build virtual ships using collaborative wide-area computer-based environments. However, modeling and simulation tools and techniques have not yet been proposed to support the acquisition and development of the software systems needed to make the overall ship system operational and effective. Thus, we propose to fill this gap with the VISTA approach.

"[VISTA] seeks to build on knowledge of best practices in "as-is" acquisition and development processes, as well as moving toward a re-tooled and reengineered "to-be" software systems acquisition and development process."

We believe that tools, techniques, and concepts embodied in the VISTA approach merit consideration and application in forthcoming large-scale system acquisitions. These include incremental acquisition interleaved with development, virtual prototyping, wide-area collaboratories, and software requirement negotiation and validation environments. However, it would be misleading to indicate that they are being used together in the manner we suggest. The VISTA approach needs to be experimentally applied and refined. Accordingly, a research and development technology road map was presented that lays out a path for the iterative, incremental evolution and integration of the technologies needed to support the VISTA vision. The technologies needed to support the VISTA approach need to be brought together and made accessible to different acquisition participants.

The VISTA approach we present is a vision of how the acquisition of software-intensive systems can be designed and streamlined for use in the years ahead. Major system acquisition programs such as the SC–21 battleships or Joint Strike Fighter aircraft are representative candidates for the VISTA approach. The success of programs such as these will depend in part on the successful acquisition and development of the software systems that enable these platforms to do their job. VISTA represents a substantial department from, and alternative to, present software system acquisition practices (STSC, 1995; SPMN, 1997). Nonetheless, we have cast it in a manner that shows how to incrementally transition from the technology and organizational practices that today support software system acquisition to the VISTA approach we envision.

Finally, moving to adopt and practice VISTA-based system acquisitions is not without its risks. Accordingly, we have sought to identify the technological and organizational transitions that must be researched, modeled, and simulated to help reduce the risks and improve our understanding of how to evolve system acquisition practices and support environments to help see the way to VISTA. In this sense, the VISTA approach could be demonstrated by applying it to the acquisition and development of a software system that incorporates the concepts in this paper and related reports (Boehm and Scacchi, 1996).

ACKNOWLEDGMENTS

Preparation of this article was supported by grants from the Air Force Rome Laboratory and the Deputy Assistant Secretary of the Air Force for Computers, Communications, and Support Systems, under contract F30602–94–C–0195, and the Office of Naval Research (ONR), under contract N00014–94–1–0889. None of the material in this report should be construed as a statement of policy, procedure, or endorsement by the ONR, U.S. Air Force, U.S. Navy, or any other U.S. government agency.

REFERENCES

Bashein, B. J., Markus, M. L., & Riley, P. (1994). Preconditions for BPR success: And how to prevent failures. *Information Systems Management,* 7–13.

Boehm, B., Bose, P., Horowitz, E., & Lee, M. J. (1995, April). Software requirements negotiation and renegotiation aids: A theory-w based spiral approach. *Proceedings of the 17th International Conference on Software Engineering,* Seattle, WA.

Boehm, B., & Scacchi, W. (1996, March). *Simulation and modeling for software acquisition (SAMSA),* final report, Center for Software Engineering, University of Southern California, Los Angeles, CA. (HYPERLINK "http://sunset.usc.edu/SAMSA/samcover.html"http://sunset.usc.edu/SAMSA/samcover.html).

Cothran, J. (1996, Winter). Battle labs: Tools and scope. *Acquisition Review Quarterly.*

Dutton, W. H., & Kraemer, K. L. (1985). *Modeling as negotiating: The political dynamics of computer models in the policy process.* Norwood, NJ: Ablex.

Garcia, A. B., Gocke, R. P. Jr., & Johnson, N. P. Jr. (1994, March). *Virtual prototyping: Concept to production.* Fort Belvoir, VA: Defense Systems Management College Press.

General Accounting Office. (1995). *Defense weapons systems acquisition* (Report GAO/HR–95–4). Washington, DC: Government Printing Office.

General Accounting Office. (1997). *Air traffic control—immature software acquisition processes increase FAA system acquisition risks,* (Report GAO/AIMD–97–47).

Haimes, Y. Y., Schooff, R. M., & Chittister, C. G. (1997, Winter). A holistic management framework for software acquisition. *Acquisition Review Quarterly.*

Kouzes, R. T., Myers, J. D., & Wulf, W. A. (1996, August). Collaboratories—Doing science on the internet. *Computer, 29*(8):40–48.

Mi, P., & Scacchi, W. (1990). A knowledge-based environment for modeling and simulating software engineering processes. *IEEE Trans. Knowledge and Data Engineering, 2*(3):283–294.

Mi, P., & Scacchi, W. (1996). A meta-model for formulating knowledge-based models of software development. *Decision Support Systems, 17*(3):313–330. (HYPERLINK "http://www.usc.edu/dept/ATRIUM/Papers/Process_Meta_Model.ps"http://www.usc.edu/dept/ATRIUM/Papers/Process_Meta_Model.ps).

Nissen, M. E. (1997, Winter). Reengineering the RFP process through knowledge-based systems. *Acquisition Review Quarterly, 4*(1):87–100.

Noll, J., & Scacchi, W. (1991, December). Integrated diverse information repositories: A distributed hypertext approach. *Computer, 24*(12):38–45.

Rechtin, W. (1991). *System architecting: Creating and building complex systems.* Englewood Cliffs, NJ: Prentice-Hall.

Scacchi, W., & Mi, P. (1997). Process life cycle engineering: A knowledge-based approach and environment. *Intern. J. Intelligent Systems in Accounting, Finance, and Management, 6*(1):83–107. (HYPERLINK "http://www.usc.edu/dept/ATRIUM/Papers/Process_Life_Cycle.html"http://www.usc.edu/dept/ATRIUM/Papers/Process_Life_Cycle.html).

Scacchi, W., & Noll, J. (1997, September-October). Process-driven intranets: Life-cycle support for process reengineering. *IEEE Internet Computing, 1*(5):42–49.

Scacchi, W., Noll, J., Knight, C., & Miller, F. J. (1997, May). *(Re)Engineering research grants management: From acquisition reform to knowledge brokering at ONR.* Paper presented at the NSF Workshop on Research and Development Opportunities for Federal Information Services, Arlington, VA. (HYPERLINK "http://www.usc.edu/dept/ATRIUM/NSF–FIS–Workshop.html"http://www.usc.edu/dept/ATRIUM/NSF–FIS–Workshop.html).

Schorr, H., & Stolfo, S. (1997, June). *Towards the digital government of the 21st century,* final report, NSF Workshop on Research and Development Opportunities for Federal Information Services. (HYPERLINK "http://www.isi.edu/nsf/final.html"http://www.isi.edu/nsf/final.html).

Smith, B. C. (1996). Limits of correctness in computers. In R. Kling (Ed.), *Computerization and controversy* (pp. 810–825). New York: Academic Press.

SC–21 information system (SC-21). (1997). (HYPERLINK "http://sc21.crane.navy.mil/"http://sc21.crane.navy.mil).

Software Program Managers Network (SPMN). (1997, October). *The condensed guide to software acquisition best practices.* Available from SPMN (HYPERLINK“http://www.spmn.com/products.html”http://www.spmn.com/products.html).

Software Technology Support Center (STSC). (1995, February). *Guidelines for successful acquisition and management of software-intensive systems: Weapon systems, command and control systems, management information systems. Vols. 1 & 2.* Washington, DC: Department of the Air Force.

Wilson, J. (1996, Fall). Battle labs: What are they, where are they going? *Acquisition Review Quarterly.*