Security Engineering in an Evolutionary Acquisition Environment[1]

Marshall D. Abrams
The MITRE Corporation
1820 Dolley Madison Blvd.
McLean, VA 22102
703-883-6938
abrams@mitre.org

ABSTRACT

This paper describes the integration of the Systems Security Engineering (SSE) process into the Evolutionary Acquisition (EA) strategy. Spurred by inadequacies of traditional acquisition strategies, large system acquisitions are using EA in the Department of Defense and the Federal Aviation Administration (FAA); this paper focuses on the FAA. Following an introduction to EA, the waterfall and spiral system development models are presented as background. Security engineering is developed as a special case of the spiral model and integrated with EA.

1. INTRODUCTION

In this paper "acquisition process" is defined to include the development and deployment of the result. "Acquisition" by itself implies activities leading up to contract award. While the duration of the acquisition itself typically exceeds the technology and requirement cycle by a multiple of two to three, this paper is not restricted to that portion of the problem.

Experience with system acquisition has shown that the use of conventional Government acquisition strategies has often led to unsatisfactory results. Participants in the commercial acquisition process have not followed the same path. They have "outsourced" the service, in general, after their internal information resources management shops fail to achieve the specific goals. The principal difficulties with traditional Government acquisition activities have been:

- Imbalance between duration of the acquisition and changes in the environment
  - The time required to complete the acquisition has been too long.
  - Changes in requirements and technology have overtaken the acquisition process.

- Immature and evolving requirements and immature and evolving software development tools
- Entrenched legacy systems, established bureaucracy, and existing culture to new technology and ways of doing business
  - Often the new technology cannot do the same job as fast, cheaply, or as familiar as the old application)
  - Business process re-engineering may be necessary for new technology acceptance.

EA has been developed in response to this challenge. An introduction to EA in weapons systems acquisition may be found in (JLC, 1995). EA is part of a revised acquisition strategy. EA is the salient characteristic for the purposes of this paper and is, therefore, used as the identifier for the larger acquisition strategy (AFMC, 1998). Other aspects include the use of commercial off the shelf (COTS) products, performance based specifications, cost as an independent variable (CAIV), and project risk management. The Defense Department 5000 series of acquisition documents has been revised.

The planned evolution of the National Airspace System (NAS) (FAA, 1996) fits the EA concept very well. NAS, an international system-of-systems, is envisioned to evolve over a 20-year period from dependence on proprietary and obsolete hardware and software toward open systems and COTS hardware and software. Legacy components will be replaced as system requirements evolve and budgetary constraints permit. NAS operational requirements are evolving in such areas as free flight[2] and transition from

[1] The contents of this material reflect the views of the author. Neither the Federal Aviation Administration nor the Department of Transportation makes any warranty or guarantee, or promise, expressed or implied, concerning the content or accuracy of the views expressed herein.

[2] *Free flight* describes aircraft operations that allow pilots more autonomy and independence while operating aircraft than is permitted today in maneuvering their aircraft without the prior approval of air traffic controllers (RTCA, 1995).

terrestrial navigation aids to those based on the Global Positioning System (Dana, 1995). It is anticipated that the NAS vision will continue to evolve. The requirements for the next year or two are in sharp focus while the vision of the far-term is seen only as meeting general objectives.

The security requirements are similarly evolving. The legacy systems relied on "security through obscurity," predicated on a threat model focused on low-resourced malicious activity. Executive Order 13010, recognizing contemporary concerns for integrity and availability of the National Information Infrastructure (NII)—of which NAS is a recognized component—has stimulated the FAA to revise its information security posture. The shift to COTS platforms and public protocols, and the emergence of a potential well-resourced terrorist threat also contribute new vulnerability elements driving the pace of the evolving security requirements.

The purpose of this paper is to present acquisition/system development and security engineering models as a way of understanding, examining, and managing change. All models involve abstraction, simplification, and suppression of detail. Nevertheless, they are useful in describing major paradigm shifts, such as those described in this paper.

Security engineering involves concerns such as protecting information and information technology systems from unauthorized policy violations to integrity, availability, and confidentiality. Information systems security (INFOSEC) encompasses the automated elements of such concern.

Following an overview of evolutionary acquisition and system development models, the paper introduces the system security engineering model and practice steps of understanding context, analyzing security risks, developing a plan, developing the system, and managing the subsequent release. The beginnings of a case study are introduced and future studies identified.

2. EVOLUTIONARY ACQUISITION MODEL

The EA environment incorporates a strategy for system development and use when achievement of the desired overall capability will require the system to evolve during its development, manufacture, or deployment. EA accommodates both the lack of COTS technology and evolving requirements as the users get their hands on the system and evolve new expectations. EA has evolved from the Pre-Planned Product Improvement (P3I) acquisition strategy. The major approach that underlies EA is the encouragement of early fielding of a well-defined core capability in response to a validated requirement that is expected to persist over a significant portion of the evolution.

Incremental releases providing additional or revised core capabilities will then follow, the requirements for which will result from (1) continuous feedback from the developer, independent testing, and the user; and (2) application of desirable technology within the constraints of time, requirements, cost, and risk. Many of the following characteristics and objectives of EA were formally espoused in an AFCEA report (1982) and have evolved since then. The basic definitions and core concepts have been maintained.

EA is particularly appropriate to most projects of large size, high complexity or long duration. Often the individuals who documented the requirements are no longer around when the design is reviewed, and the design reviewers aren't there for the deployment. Experience indicates that change to one or more of the following factors is likely to surface during system definition and development:

- Requirements and technical uncertainties
- Funding availability
- Technology
- Scheduling
- Interoperability and commonality requirements
- The need in some kinds of systems for continuous user involvement
- Instabilities due to environmental constraints

When any of the following conditions are true, the use of an EA strategy is indicated:

- Complete requirements cannot be adequately stated at the initiation of complex programs.
- Requirements are expected to change frequently over the life of the program.
- Users cannot specify acceptability criteria in advance due to their subjective nature.

When technological uncertainties exist, an evolutionary approach may be adopted even when the desired goal can be rather precisely defined and achievement can be objectively measured. EA has been identified as a means of dealing with the following considerations:

- Risk associated with technological uncertainties
- Necessity to incorporate research results
- Prolonged development periods
- Complexity of very large systems
- Changes occurring at the limits of a state-of-the-art technology

When requirement uncertainties indicate an EA approach, the program may involve little or no advance development. The FAA has adapted a "build a little, field a little" approach. For programs with requirements uncertainties, succeeding blocks of work after completing the first block cannot be adequately specified until feedback from some user, technology, or policy maker is received on the usefulness of the solution and modifications which are perceived as needed.

The objectives of EA as stated as follows:

- Develop a vision of the functional capability desired for the full system. Some definitions identify the lack of specificity and detail in describing the final system capability as the distinguishing characteristic of EA.
- Provide a concise statement of operational concepts for the full system.
- Develop a flexible, well-planned overall architecture that includes process for change. This process for change allows the system to be designed and implemented in an incremental fashion with minimum regression testing. The following recent advances are enabling change at reasonable cost and impact:
    - Open systems
    - Domain architectures
    - Software development tools
- Develop a plan for incrementally achieving the desired total capability that adheres to life-cycle cost effectiveness.
- Construct mutually consistent definitions for each increment among concept of operations, requirements, and architecture
- Ensure early definition, funding, development, testing, fielding, supporting, and operational evaluation of an initial operational core capability.
- Promote continual dialogue and feedback among users, developers, supporters, and testers.

EA heightens, rather than obviates, the need for early program planning and engineering activity. Significant effort is required early in the program to permit adopting a strategy that accommodates change. Proper process control must be provided. It is important to define the developmental increments and the system performance that should be achieved. Doing so will provide a basis for review of changing functional requirements that appear during the development process.

To retain control and show discrete progress, EA allows changes in requirements between incremental releases.[3] Allowing requirements to change during the construction of an increment can be costly in time, dollars, and performance. Completion of a release may be marked by establishing a new baseline. There is a varying level of requirements control. Requirements are accumulated against a planned release and then frozen to stabilize content of that release. Requirements which are outside the frozen baseline for a release are queued for a subsequent release. Additionally, filtering of user feedback is necessary to prevent requirement whiplash (I liked the old one better, I want a new one, etc.).

Changes to functional requirements (especially additions to current requirements) can be controlled within each incremental release by accepting only customer or user-designated high-priority changes. It is also necessary to collect lower priority change requests and select these for accomplishment concurrent with related high priority changes. EA gives one a better feel for how to solve the high-priority issues and gives better long-term statistics on achievement. Crucial requirement changes are identified on the basis of feedback concerning effectiveness and suitability from actual operation and maintenance of earlier releases.

The need to manage requirements change is perhaps greatest when the changes affect software in the development cycle. It is often possible to effect a performance change through a change to the system software. However, if the performance improvement affects the release schedule, then careful consideration of the tradeoff is necessary at all level of the organization. The further along the development process, the more difficult it is to make software changes and retain release integrity for test and documentation. Significant security engineering is also required as part of the integration process if the system changes affect the security requirement or solution sets.

Configuration management and full system design documentation are important for any acquisition process. In an evolutionary process, careful attention to the evolving architecture and the corresponding system increments is of paramount importance.

The EA approach may be tailored to accommodate the following characteristics:

- The degree of user and developer knowledge and involvement required
- Requirements or technological uncertainties
- The amount and complexity of required development
- Opportunities for using COTS products
- The selection of software language, technology, and development tools

3. SYSTEM DEVELOPMENT MODELS

The waterfall and spiral models of system development inspired the system security engineering present below. Accordingly, they are briefly reviewed. Thee models can be viewed as being centered on each lifecycle phase, as presented by a DoD 500 milestone.

3.1 Waterfall Model

The most common of the formal system development techniques, first developed by Royce (Royce, 1970) and popularized by Boehm (Boehm, 1976), was known as the *waterfall* model because of the way the model was depicted (see figure 1). In this model, *phases* of information system development are shown as steps cascading downward, with information flowing from one step to the next step through a formal *signoff* process.

[3] Incremental releases have also been called "builds" and "blocks".

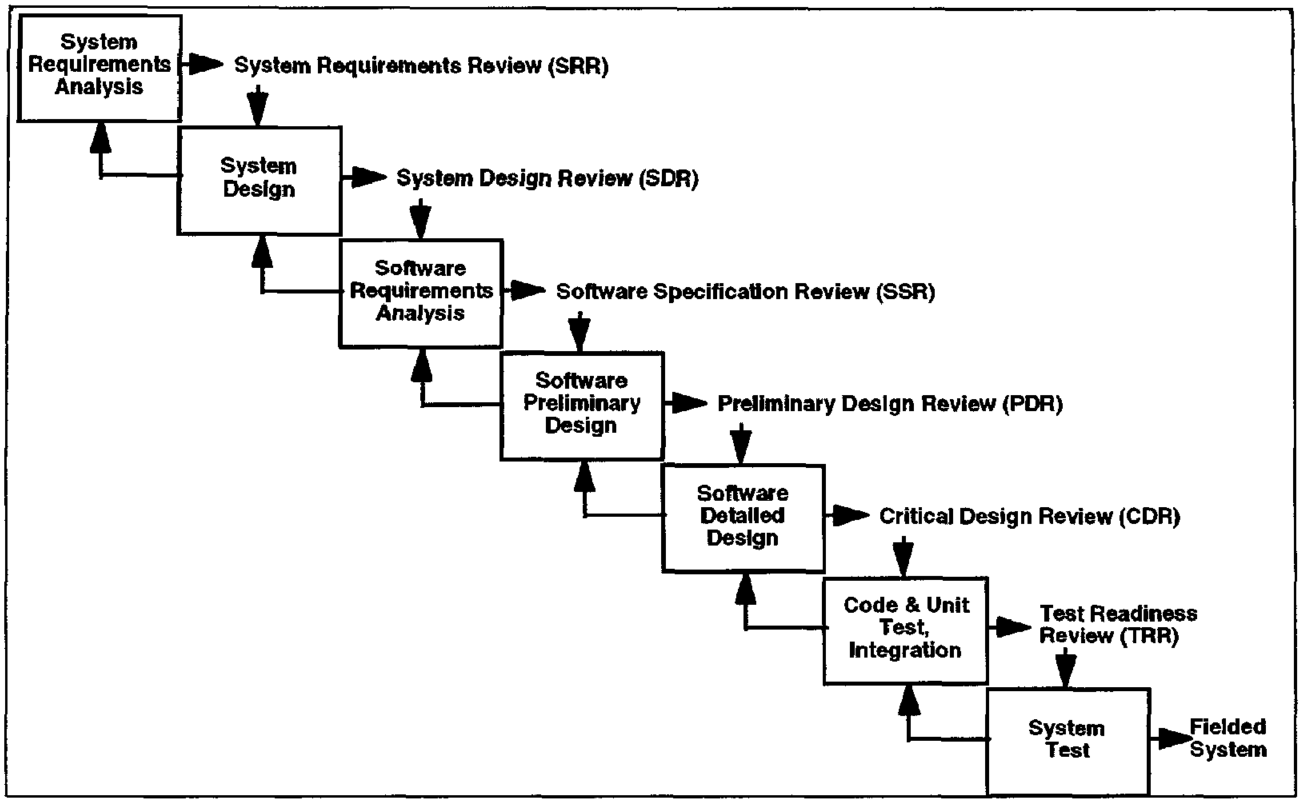
System Requirements Analysis
System Requirements Review (SRR)
System Design
System Design Review (SDR)
Software Requirements Analysis
Software Specification Review (SSR)
Software Preliminary Design
Preliminary Design Review (PDR)
Software Detailed Design
Critical Design Review (CDR)
Code & Unit Test, Integration
Test Readiness Review (TRR)
System Test
Fielded System

Figure 1. Waterfall Model of Software Development

The resulting system was expected to satisfy the end-user's needs when it was delivered as the end user had reviewed and approved its functional design. This approach was successful as long as systems could be built in a relatively short period of time (measured in terms of fractions of the technology cycle) and when such systems spanned only a single operational entity of an organization. When implementation exceeds one technological cycle, user acceptance is doubtful.

The waterfall model became the most widely known and used system development approach in the software industry. If a system being built using the waterfall model failed to meet user expectations at the time of its delivery, the failure was generally ascribed to inadequate requirements analysis or possibly to poor change control. As the information systems became more ambitious and complex, it took longer to build them, increasing the probability that some event(s) would occur to cause a change in system requirements necessitating requirements change management.

System developers also discovered that no matter how hard they tried to understand the end-user's requirements, requirements creep and defects in their understanding might result in a change in the design. There was also the risk that the end users could discover they had misunderstood the implications of a particular design solution and ask for a change that impacted some element of the system design. Or, in the process of implementing a solution, an implementer could decide that the solution was unworkable or ineffective, and decide to change the way it would be implemented. This too might result in a change to the system requirements.

Thus, despite the best practice approach using the waterfall model, some system development projects continued to fail. This paper can only hint at some of the contributing factors. Sometimes, the system requirements could not be defined with rigor at the outset, because the end user could not articulate them well. System requirements also could be difficult to define initially if the system, when implemented, might make significant changes in the way the organization did business. In these cases, system requirements were often cast in terms of how the business currently worked rather than in terms of how the business should work. Sheer size and complexity are also limiting factors. Some systems are too complex and difficult to comprehend, specify, and develop. A long development phase relative to the rate of technological change can make the system's technology out of date as soon as the system is fielded. User expectations changed during the development cycle due to techno-hype in trade journals, use of computers at home, and the general rate of technological turnover.

3.2 SPIRAL MODEL

An alternative system development model that supports evolutionary development is Barry Boehm's spiral model (Boehm, 1987) (see figure 2). To some observers the waterfall model appeared to focus on a linear-bounded process while the spiral model emphasized cyclic recurring activities. Others observe that the waterfall can be rolled up

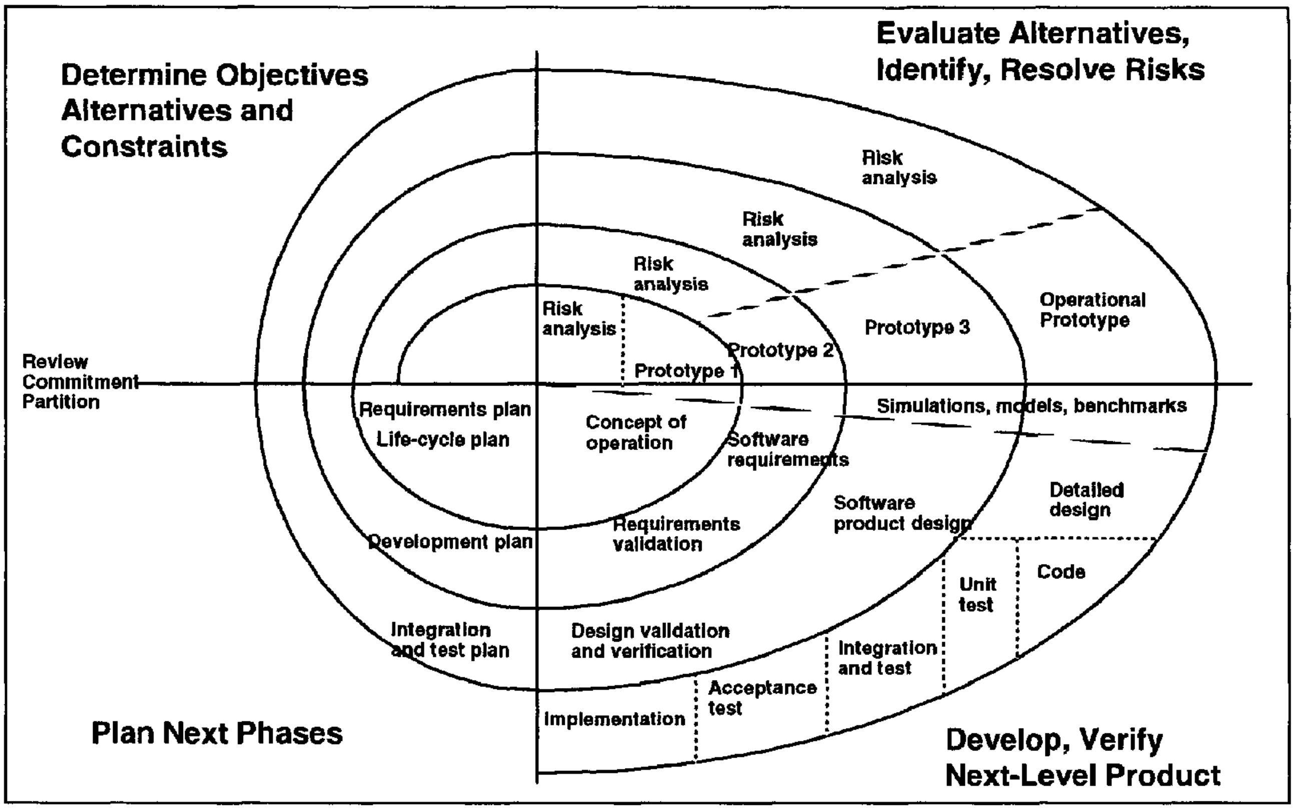

Figure 2. Spiral Model of Software Development

or the spiral can be unrolled, thereby mapping one model to the other. Boehm saw the spiral model as risk- driven because the activities at any particular point in the process are determined by the need to identify and resolve risks. In this paper we focus on the recurring security engineering activities using the spiral form as they apply to EA.

From an SSE solutions perspective, a potential vulnerability is anything that could reduce integrity, availability, confidentiality, or other SSE objectives. The countermeasures applied to reduce or eliminate vulnerability need not be specific to SSE. For example, procedural approaches and countermeasures based on safety engineering may be effective and sufficient for particular SSE vulnerabilities. Security engineering shares certain concerns and countermeasures with safety engineering. It is efficient to capitalize on the overlap. Attempts to draw a sharp distinction between security and safety have not been productive. The interested reader is referred to (Burns, 1992).

Security engineering should be a continuing activity of the systems engineering life cycle. Security engineering findings, like many systems engineering findings, are reviewed and revised, as appropriate, at various milestones in the system development. The milestones of either waterfall or spiral software development models are appropriate.

4. SSE MODEL AND PRACTICE

EA involves refining discipline and control throughout SSE development and maintenance. Confidence in the correspondence between the SSE requirements and their implementation is greater if SSE analysis and documentation are made an integral part of the development and maintenance activities. The thesis is that since each can be applied to the spiral model, SSE to match it must be spiral as well. Poorly controlled development and maintenance can result in a flawed implementation or an implementation that does not meet all of its SSE requirements. This lack of control, in turn, increases the risk of security failures.

Figure 3 illustrates the SSE spiral model. It is derived from aspects of a spiral systems development model, but does not presuppose that development should formally adhere to the spiral model. Representative activities are shown inside the spiral. Milestones and decision points mark the transitions between SSE stages. The five stages of SSE, described below, are performed for each EA cycle.

4.1 UNDERSTAND CONTEXT

The first stage in SSE engineering is attempting to understand the business process and system context. In the first iteration this knowledge will probably be highly imperfect. Future iterations will refine the understanding by virtue of experience. Anticipation of multiple iterations reduces the pressure to place premature emphasis on assuring a perfect understanding or getting the customer to

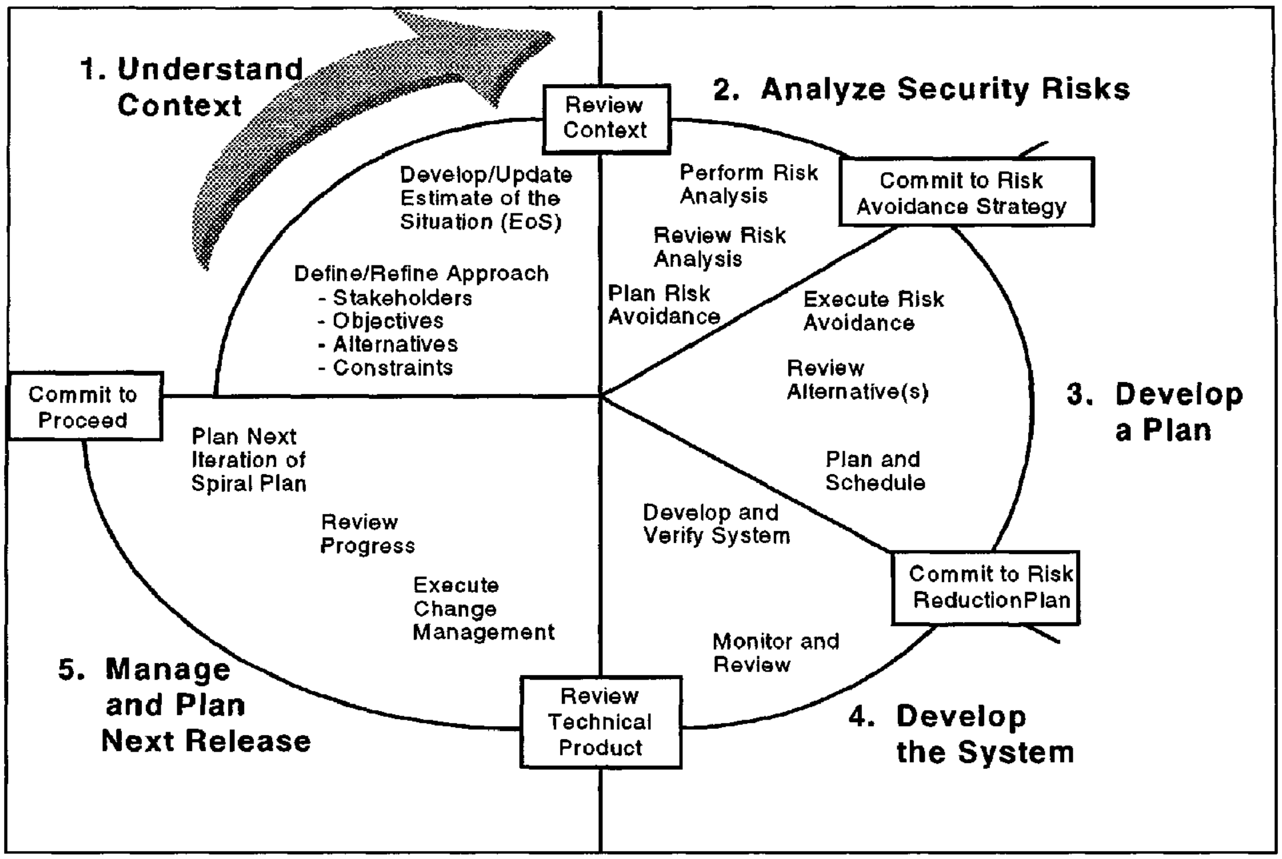

Figure 3. SSE Spiral Model

agree. The business process and system context does not exist in a vacuum. Stakeholders, objectives, alternatives, and business constraints must be identified and incorporated into the understanding. The first stage culminates in a summary review of the context.

4.2 ANALYZE SECURITY RISKS

Risk analysis occurs in the second stage. Security threat is an agency of risk in addition to the business risks addressed in EA. This stage often involves considerable judgment in identifying the nature of the security threat and the probability of that threat materializing. The idea is to establish a level of security for all information systems that is commensurate with the risk and magnitude of the harm resulting from the loss, misuse, or unauthorized access to or modification of the information contained in these information systems (OMB, 1996). With the establishment of this level of security, preparation of formal risk analyses is no longer required. OMB guidance concerning risk analysis is well adapted to EA:

> In the past, substantial resources have been expended doing complex analyses of specific risks to systems, with limited tangible benefit in terms of improved security for the systems. Rather than continue to try to precisely measure risk, security efforts are better served by generally assessing risks and taking actions to manage them.
>
> While formal risk analyses need not be performed, the need to determine adequate security will require that a risk-based approach be used. This risk assessment approach should include a consideration of the major factors in risk management: the value of the system or application, threats, vulnerabilities, and the effectiveness of current or proposed safeguards (OMB, 1996).

The second stage culminates in a documented risk management strategy.

4.3 DEVELOP A PLAN

Having determined the risk environment, the third stage addresses a plan for risk management. A 1994 Joint Security Commission report best describes the process:

> In the past, [in certain environments] most security decisions have been linked one way or another to assumptions about threats. These assumptions frequently postulated an all-knowing, highly competent enemy. Against this danger, we have striven to avoid security risks by maximizing our defenses and minimizing our vulnerabilities. Today's threats are more diffuse, multifaceted, and dynamic. We also know that some vulnerabilities can never be eliminated fully nor would the costs and benefits warrant

trying.... In most cases it is possible to balance the risk of loss or damage of disclosure against the costs of countermeasures. We can then select a mix that provides adequate protection without excessive cost in dollars and without impeding the efficient flow of information to those who require ready access to it.

The third stage produces the specifications and a risk management plan.

4.4 DEVELOP THE SYSTEM

The fourth stage implements the plan by developing the system. Note that the products of figure 2 correspond to the system of figure 3. The system is then implemented, tested, and verified. This stage exhibits major challenges to the practice of SSE. Integrating secure products into secure systems is a formidable challenge in conventional acquisition (Gambel, 1995).

EA, with its emphasis on COTS product use, exacerbates the SSE integration problem. Security designers have attempted to implement a modular solution, first building trusted operating systems, and then adding trusted networks, trusted database systems, and trusted applications. On top of this layering, user applications which have no basis for trust were integrated. In many cases, the term "Trusted" was applied, not because the confidence was high, but because the privilege set required that terminology to be used. The result was low confidence "trusted" system solutions.

This modular solution has failed (JSC, 1994). The business part of the failure was the inability to deliver the market demand for trusted products. The strategy did not represent an unavoidable commitment on the part of the DoD and Intelligence Community. Rather, it expressed the aspirations of the computer security core group. Producers failed to reap the rewards of selling evaluated products in sufficient quantity to recover the costs of participation in the program.

The technical part of the failure was the incompatibility among trusted products. Each product used similar but different security policy models and represented them in a different structure. Assumptions and design decisions were different and were often unavailable. Some evaluation reports contained censored documentation. The system integrator might be able to reverse engineer the products to determine and revamp their models and structures, but design decisions remained as potential points of failure. This reverse engineering and revamping severely reduced the value of prior evaluations. Integrators were required to encapsulate the security solution in a manner that did not impact their systems solution, that is, independent of whether security works or fails, the system will work.

The practical alternative is the application of correctness methods from the security engineer's toolkit, represented in table 1, includes formal methods, simulation, testing, process modeling, structured programming, use of computer-aided software engineering (CASE) tools, object-oriented design, and reuse of existing code. The following four interrelated strategies, evolved from recommendation by Abrams and Zelkowitz (1995), recognize that SSE is an empirical discipline:

- Evaluation of process, personnel, and abilities to identify and reinforce what is working well
- Thorough review and analysis by qualified independent security professionals of intermediate products during development with sufficient time and resources allocated to correct deficiencies
- Rigorous testing based on the assertions not specifically validated as having been corrected by the preceding analysis
- Recognition of critical milestones in system development. This recognition includes understanding of risk of failure and conducting a cost/benefit analysis for reducing this risk further

Table 1 presents representative security engineering methods and allocates them to the stages of the security engineering spiral model. The selection of methods and the table entries are not rigorously derived; they are presented for illustrative purposes only. Completing such a table for a specific cycle of a particular system is only one part of the overall security engineering process. The fourth stage culminates in a review of the technical product.

Consider, for example, the testing row of table 1. Testing occurs during the system development stage. The skills required to conduct the tests (as differentiated from test design) are readily available. The cost of testing and the cost-effectiveness are average among the methods. Testing is well suited to the evaluation of complex systems.

4.5 MANAGE AND PLAN NEXT RELEASE

The fifth stage builds on the four previous stages, reviewing progress, revising the plan in light of experience, and making plans for the next cycle's incremental release. Change management is an important technique for completion of a cycle. During the conduct of the cycle, many ideas for improvements will undoubtedly occur. Although difficult to resist the temptation to implement the best ideas, prudence dictates that plans and objectives be held as constant as possible for one cycle.

To be most effective in an EA environment security engineering must also adapt to near-sighted focus. High-level long-term security objectives probably change little and slowly while short-term strategies and mechanisms will be more dynamic. Similarly, changes in risk, threat, and vulnerability are more in focus the more immediate they are. During this stage the security engineer should reconfirm the long-term strategy and plan the tactical solutions for the next cycle.

Method	When Used	Skill Required	Cost	Cost Effectiveness	Applicable to Complexity
Formal Methods	2, 3	▲	▲	▼	▼
Simulation	1, 2, 3	▲	▲	▼	▼
Testing	4	▼	■	■	▲
Structured Programming	4	▼	▼	■	▲
CASE Tools	4	▲	▼	▼	▲
Object-Oriented Methods*	3, 4	■	■	▼	▲
Code Reuse	4	▲	■	■	▲

Table 1. Characteristics of Correctness Methods

Key:

Security Engineering Stages

1. Understand Context
2. Analyze Risks
3. Develop Plan
4. Develop System

Skill Rating

▲ above average, rare

■ average

▼ below average, common

* Object-oriented methods and code reuse are functions of the software development tools, language, and architecture selected.

Reviewing progress includes ascertaining the extent to which the objectives have been achieved. Progress is reviewed against both old and new objectives. The view backward can be used as input to process improvement as well as reward to the implementers. In an EA environment we expect that the target may change. Separating the measurement of achievement from change in plans is important.

The fifth stage produces a commitment to proceed based on an understanding of past accomplishments and the refined view of the future inherent to EA.

5. BEGINNING A CASE STUDY

The NAS Infrastructure Management System (NIMS) acquisition, an ongoing example of EA with an SSE component, is a test case for the approach described in this paper.

One of the first steps for NIMS security is creation of a reference security architecture, i.e., an architecture that is not implementation specific. Eventually, the security architecture will also need to address security from the functional and application perspective.

This reference security architecture provides a generic technical reference model of security that is "neutral" in terms of the specific implementation details. It allows flexibility in the design and development of INFOSEC in different phases of the NIMS system life cycle. The reference security architecture lends itself to open systems and industry standards to minimize life-cycle costs, maximizes the use of COTS hardware and software solutions, and supports incremental development of functions. The reference security architecture supports incremental changes as the design moves toward alignment with the client's system architecture. The goal of the reference security architecture strategy is to mitigate controllable risk of security failures during the EA life cycle. It will hopefully contribute to a reduction in costs by minimizing large-scale security changes. It is also intended to support security for the distribution of processing across the different nodes that provide centralized management of the client system infrastructure.

In summary, the reference security architecture has the following features

- Allows for extensions that support new functionality, technology, devices, applications, information, and capacity
- Allows maximum use of COTS security products and is not dependent on any particular vendor or specific system design
- Will evolve with advances in security technology

- Allows flexibility and scalability to accommodate changes in requirements and to allow rapid integration of new capabilities and technologies
- Is consistent with the *Information Technology Security Evaluation Common Criteria*, version 1.0 (1996); the *Office of Management and Budget (OMB) Risk Management Paradigm*, OMB Circular A-130 (1996); and the *President's Commission on Critical Infrastructure Protection*, Executive Order 13010 (1996)

The reference security architecture for computing resources is an architectural view of security for a distributed computing system. The reference security architecture addresses both hardware and software in the distributed network. Security services include:

- Authentication services provide a means for one entity to validate the identity of another.
- Secure access management services protect the resources of messages and the data-handling network from unauthorized use. Once a user has been identified and validated using the authentication service, access management ensures that the user receives or modifies only that information that he/she is authorized to have.
- Data confidentiality services as needed ensure that the content of messages or data is known only to the sender or user and the intended recipients. They also include a service to protect against traffic flow analysis attacks.
- Auditing services maintain a log of security-related activities, including successful logons, failed logons, failed authentication attempts, and encryption/decryption events.
- Data integrity services protect against the modification of message and data contents.

A primary objective of the reference security architecture is to provide security management of the infrastructure, which will consolidate a large number of facilities currently performing infrastructure monitoring and control. This does not, however, dictate that security management resources must be centralized or that all functions must reside at centralized management facilities.

NIMS will provide the flexibility to distribute the processing of management information, while providing an integrated view of the system infrastructure status and centralized monitor and control capabilities at centralized operational control centers. Distributed services are used within and between operational control centers.

Three possible systems that may exist at operational control centers will require security protection:

- A core system server that hosts fundamental system-related server services
- A system that has the management application of Operational Configuration Management (OCM)
- A workstation

In addition, the reference security architecture protects peripherals that attach directly to corresponding hardware or are attached to the local area network (LAN) within the command node.

NIMS will provide comprehensive, precise and timely management information. The system is based on industry and international standards for information exchange, extended as necessary for information unique to NIMS. Database servers will be located at each operations control center and at selected locations of high system management activity (e.g., large work centers). Object-oriented or relational database technology will be used to implement the databases. The NIMS information architecture is a seamless, distributed information domain shared by the operational control centers, and multiple work centers and other system management facilities. External information systems may be sources and/or receivers of data from NIMS.

6. FUTURE STUDIES

This paper has introduced a model for SSE is a reasonable framework for further development. In data-centric systems, emphasis is placed on protecting data from software. It has been suggested that EA systems tend to not be data-centric, but functionality-centric. Software doesn't attack the assets; software and data *are* the assets. The Joint Security Commission has said that the old strategy has failed. We need to explore how security engineering in an EA environment may be part of a successful new strategy.

In order to be made more concrete, the concepts presented in this paper need to be applied to real systems. In addition to the ongoing NIMS acquisition, it would be valuable to reexamine recent system developments from the perspective of this paper in order to refine the methodology and generate detailed procedures and processes. We must identify key parameters and issues within the process that impact success from both EA and SSE perspectives. An initial set of key parameters includes the following:

- Product development versus use of COTS products
- Frequency, amount, and complexity of changes
- Quality of specifications
- Key personnel competency and stability

7. SUMMARY

Evolutionary acquisition techniques are being used for large system development efforts to reduce the time required to complete a system relative to the rate of changes in requirements and technology. The practice of security engineering can adapt to EA by following a variant of the spiral systems development model applied to each incremental release. SSE is one of the continuing activities of the systems engineering life cycle.

The probability of success is greater if security engineering analysis and documentation are made an integral part of the development and maintenance activities. The security engineering spiral model is presented as having five stages: understanding the context, analyzing security risks, developing a plan, developing the system, and managing and planning subsequent release(s).

8. REFERENCES

[1] Abrams, M. D., and M. V. Zelkowitz, December 1995, "Striving for Correctness," *Computers & Security*, Vol. 14, No. 7, pp. 719 - 738, Elsevier Advanced Technology, Oxford, UK.

[2] Armed Forces Communications and Electronics Association (AFCEA), 1 September 1982, *Command And Control (C^2) Systems Acquisition Study Final Report*, Falls Church, VA.

[3] Air Force Material Command, 1998, Acquisition Strategy, http://www.afmcwpafb.af.mil/organizations/HQ-AFMC/DR/drihome/acqref/aqustrat/aqstrat.htm

[4] Boehm, B. W., 12 December 1976, "Software Engineering," *IEEE Transactions on Computers*, Vol. C-25, No. 12, pp. 1226-1241.

[5] Boehm, B. W., 1987, "A Spiral Model of Software Development and Enhancement," *Software Engineering Project Management*, IEEE, Inc., pp. 128-142.

[6] Burns, A., J. McDermid, and J. Dobson, 1992, "On the Meaning of Safety and Security," *The Computer Journal*, Vol. 35, No. 1, pp. 1-15.

[7] Dana, P. H., 1995, *Global Positioning System Overview*, Department of Geography, University of Texas at Austin, Texas and at http://www.utexas.edu/depts/grg/gcraft/notes/gps/gps.html

[8] Federal Aviation Administration, October 1996, *National Airspace System Architecture*, Version 2.0 (or most recent version), Office of System Architecture and Program Evaluation (ASD), Washington, DC.

[9] Gambel, D., and J. Hemenway, October 1995, "The Use of Generic Architectures in System Integration," *Proceedings of the 18th National Computer Security Conference*, pp. 431-446.

[10] Joint Logistics Commanders (JLC), May 1995, *Guidance for Use of Evolutionary Acquisition Strategy To Acquire Weapon Systems*, Defense Systems Management College Press, Fort Belvoir, VA.

[11] Joint Security Commission (JSC), 28 February 1994, *Redefining Security—A Report to the Secretary of Defense and the Director of Central Intelligence*, Washington, DC.

[12] Office of Management and Budget (OMB), 8 February 1996, *Management of Federal Information Resources (Transmittal Memorandum No. 3)*, Circular A-130, Washington, DC.

[13] President of the United States, 15 July 1996, *Critical Infrastructure Protection*, Executive Order 13010,Federal Register.

[14] Royce, W. W., August 1970, "Managing the Development of Large Software Systems: Concepts and Techniques," *Proceedings, IEEE WESCON*, pp. 1-9.

[15] RTCA, Incorporated, January 1995, *Report of the RTCA Board of Directors' Select Committee on Free Flight*.