

Journal of Homeland Security and Emergency Management

Volume 3, Issue 4

2006

Article 3

Cybersecurity: From Ad Hoc Patching to Lifecycle of Software Engineering

Clyde G. Chittister*

Yacov Y. Haimes†

*Carnegie Mellon University, cc@sei.cmu.edu

†University of Virginia, haimes@virginia.edu

Cybersecurity: From Ad Hoc Patching to Lifecycle of Software Engineering*

Clyde G. Chittister and Yacov Y. Haimes

Abstract

The role of information assurance (IA) is critical for cyber-based technologies and products, and the risk of cyberterrorism to IA is omnipresent. In particular, to achieve IA, young and dynamic developing technologies and products should be using a defined lifecycle that leverages and builds (throughout the developmental lifecycle) on a rich and proven body of knowledge and practices in risk assessment and management. The lifecycle of software development must include the following (not necessarily sequentially): the needs and requirements; specifications; contractor selection; conceptual design; systems integration, demonstration, and validation; engineering manufacturing, development, and production; and maintenance and major upgrade. In addition to addressing the functionality of the lifecycle development, from the risk analysis perspective it is just as important to focus on (1) the people's perspectives—namely, the individual, the team, the management, and the stakeholder, (2) the hardware-software perspectives, especially the risks associated with the commercial-off-the-shelf (COTS) products and (3) the environment within which the entire system operates. This paper follows and builds on two papers previously published in this journal on the risks of terrorism associated with supervisory control and data acquisition (SCADA) and other cyberdependent systems. Its thesis is that the reliability and integrity of such systems, and thus, the corresponding interdependent infrastructures served by them, are contingent on the following three principles of IA and cybersecurity. Adhering to these principles can be instrumental in achieving the desired level of IA and cybersecurity:

- (1) Risk of software intrusion must be assessed and managed throughout the lifecycle of software development, focusing on both the functionality of software development and on the people involved in the process, knowing that hackers will exploit every weakness in the system.
- (2) Achieving information assurance and cybersecurity must be placed high on the priority list of top management. (The two are intricately dependent on software quality and telecommunications fidelity). This is synonymous with performing a holistic risk assessment and management.
- (3) Risk management of cyberterrorism must be the domain priority of the entire development

*The principles advocated in this paper were inspired by the work of Barry Boehm, whose contributions and long-term association with the Software Engineering Institute are much appreciated. We also thank Grace Zisk for her technical editorial help. CMM, CMMI, ATAM, and Capability Maturity Model are registered in the US Patent and Trademark Office by Carnegie Mellon University. SM TSP is a service mark of Carnegie Mellon University.

team and the organization's management. It must be achieved from the perspectives of the total system throughout the software and system development's lifecycles.

Building on the multifarious sources of risk envisioned during the lifecycle of software development through Hierarchical Holographic Modeling, resilience in cybersecurity through risk management is discussed. The human role in IA and cybersecurity and the centrality of the educational dimension in risk management are also introduced.

KEYWORDS: cybersecurity, risk management, cyberterrorism

Preface

Two publications have inspired the research presented in this paper. The first is the special publication *Processes for Producing Secure Software: Summary of US National Cybersecurity Summit Subgroup Report* [Davis et al. 2004].

The second [Wulf and Jones 2004], *A Perspective on Cybersecurity Research in the United States*, calls for a new approach to cybersecurity that goes beyond “perimeter defense.” This paper embraces the philosophy that “circling the wagons” around software-intensive systems has not and is not likely to protect against cyberterrorism in the future.

A. Backgrounds

The use of information technology (IT) by industry and government organizations for data acquisition, process control, information management systems, and numerous other cyber-based activities, continues to grow. Supervisory control and data acquisition (SCADA) and other cyber-dependent systems are among the IT on which many industries and sectors of the economy depend and heavily rely, including oil and gas, electric power, telecommunications, transportation, and water resources.

The extensive use of SCADA-type systems has generated a major market for their production and maintenance. It also has resulted in a large number of systems with functionality common to SCADA and other cyber-dependent systems, albeit with different acronyms, including digital control systems (DCS), and computer-aided dispatch (CAD). The fundamental fact is that the reliability and integrity of all of these systems are intricately dependent on appropriate information assurance and cybersecurity. Thus, it is of central importance to achieve an acceptable level of risks to the telecommunications media on which SCADA systems are so inherently dependent and in turn to the well-being of the economy they serve. (Risk is defined as the probability and severity of adverse effects [Lowrance 1976].) This is a sequel to two papers on this theme previously published in this journal.

The first of this triad on SCADA systems [Chittister and Haimes 2004], presents a detailed discussion on the use of these systems by the railroad sector. Hierarchical holographic modeling (HHM) and control objectives for information and related technology (CobiT) are introduced and used to identify sources of risk to SCADA systems in the railroad sector. The vulnerabilities to terrorist attacks on IT, SCADA and other cyberdependent systems, global positioning systems (GPS), and satellites are explored. The second paper focuses on the fact that many sectors of the economy and other critical infrastructures are highly coupled and their interdependencies render them at risk to cyberterrorist attacks [Haimes and Chittister 2005]. This fact is further exacerbated because they are often remotely

controlled and managed through SCADA systems, which are vulnerable to such cyber intrusion. The myriad sources of risk to SCADA systems identified through HHM in the first paper serve as the impetus to a roadmap presented in the second paper for quantifying the efficacy of risk management of interdependent SCADA systems. Central to this quantification is the deployment of the Inoperability Input-Output Model (IIM) [Haines and Jiang 2001; Haines et al. 2005a; b]. This is a Leontief-based model that enables accounting for both the intra- and interconnectedness within each economic sector and infrastructure of the economy. At the core of the IIM is the notion of *risk of inoperability*, which describes a critical infrastructure's expected level of dysfunctionality. The metric used for quantifying the efficacy of risk management of interdependent SCADA and other cyberdependent systems builds on the economic losses generated by the IIM resulting from a cyber attack. It includes losses *with* and *without risk management*, and *the cost of risk management*.

B. Introduction

The balance for achieving secure information systems is tilted more toward short-term tactical measures, focusing on fire walls, patching, and response to cyber attacks, and less toward long-term, strategic approaches that address the entire software lifecycle development. There are undoubtedly many reasons for this state of affairs, the most pertinent of which for this paper is the poignant statement by Wulf and Jones [2004]: "The truth is that we don't know how to build secure information systems." Clearly this is a problem of a global magnitude—the annual cost to the US economy alone resulting from cyber attacks is estimated to surpass \$100 billion. For example, financial institutions have neither the tools with which to assess their losses due to "phishing" or "pharming," nor knowledge of the sources of hackers. Also, there is a significant gap between the state-of-the-art and the-state-of-practice in cyber protection. Based on responses from practitioners in US corporations, government agencies, financial institutions, medical institutions, and universities, the findings of the 2003 Computer Crime and Security Survey confirm that the threat from computer crime and other information security breaches continues unabated [Computer Security Institute 2003]. Does quality control constitute a large part of the problem and possibly its solution? To address this question, it is instructive to review the evolution of quality control of hardware products just two decades ago. Many manufacturers adhered to what is known as three-sigma quality control—namely, accepting defects of one in a thousand before they embraced the six-sigma metric that reduces the level of defects to about one in a million. The three orders of magnitude in product improvement, which has revolutionized the market, demanded a restructuring of processes, better integration and coordination

between users, customers, vendors, manufacturers, engineers at all levels, and most importantly, better training and institutional and organizational commitment to quality.

Clearly, the lifecycle of software development is markedly different from the manufacture of hardware products. In particular, the process of software architecture, coding, and testing markedly differs from that of hardware. Nevertheless, there are some critical elements in hardware quality control that can be embraced and adapted when reviewing the lifecycle of software development. Information security cannot be achieved on an ad hoc basis through piecemeal and uncoordinated strategic and tactical engineering and managerial decisionmaking. The imperative need for IA transcends economics, expediency, or inconvenience. Indeed, it is at the heart of trustworthiness in the computer technology that dominates every aspect of daily life, national security, and the integrity and safety of the individual identity (given the growing incidents of identity theft).

Existing guidelines and practice for software development processes do not explicitly address the imperative of information assurance within the software development process. Thus, a significant source of risk of cyber attacks and the loss of information assurance is the failure to maintain software engineering practices that do not, for example, adhere to the high quality advocated by the Capability Maturity Model® (CMM®) [Humphrey 1989] and the Capability Maturity Model Integration (CMMI®) [Chrissis et al. 2003]. The CMM “is a software process assessment framework that guides management through a five-level improvement program.” For example, “at level 5, the organization has a regular way to introduce technology improvements and to continually improve its process” [Humphrey 2002]. Although level 5 does not explicitly address information assurance, it does move software engineering development in the right direction. In particular, the CMMI goes beyond CMM by integrating three significant source models:

1. Capability Maturity Model for Software,
2. The Systems Engineering Capability Model (SECM), also known as Electronic Industries Alliance 731 (EIA 731) [EIA 1998], and
3. The Integrated Product Development Capability Maturity Model (IPD-CMM).

Even the combination of these models into a single improvement framework, which is intended for use by organizations in their pursuit of enterprise-wide process improvement, does not explicitly address the imperative of information assurance in the process of software development.

The literature on cybersecurity is replete with “what can go wrong” when hackers and unauthorized intruders gain access to a supposedly secure system. These sources of risk can be invariably attributed to the nature of software and the complexity of software architectural design. McDermid [1991] has this to say on the nature of software: "Software has no physical existence and, in general, few reliable software metrics have been used. Software may have a much greater complexity than hardware and often includes highly structured data as well as logic. It is deceptively easy to introduce changes into software, the effects of which can propagate explosively." The literature is similarly replete with myriad *tactical and short-term* risk management options whose primary focus is on the latest virus, worm, or Trojan horse.

On the bright side of the equation, Gilliam et al. [2003] provide an integrated software security checklist for the software lifecycle, calling it *Software Security Assessment Instrument*: “(1) a software security checklist; (2) a vulnerability matrix that categorizes vulnerabilities and exposure; (3) a flexible modeling framework for verification of requirements; (4) a property-based tester for testing vulnerabilities; and (5) a collection of security assessment tools.” Although this literature is important, timely, and valuable, a holistic approach is needed to reach an acceptable level of information security. Such an approach would identify and address the epistemological problems associated with cybersecurity and information assurance across the entire lifecycle of software engineering. This challenge, which addresses root cause rather than merely direct cause, is the subject of this paper. To meet it, there must be a seamless integration between software development principles and information assurance principles within a broad cybersecurity. In his book *Winning with Software*, Watts Humphrey [2002] identifies the following three *Principles of Software Management*:

- (1) *Recognize that you are in the software business.*
- (2) *Quality must be the top priority.*
- (3) *Quality software is developed by disciplined and motivated people.*

Building on both the content and structure of Humphrey’s triplet principles, this paper advocates the following three principles of *Information Assurance and Cybersecurity through Software Management*:

- (1) *Risk of software intrusion must be assessed and managed throughout the lifecycle of software development, focusing on both the functionality of software development and on the people involved in the process, knowing that hackers will exploit every weakness in the system.*

- (2) *Achieving information assurance and cybersecurity must be placed high on the priority list of top management. {The two are intricately dependent on software quality and telecommunications fidelity}. This is synonymous with performing a holistic risk assessment and management.*
- (3) *Risk management of cyberterrorism must be the domain priority of the entire development team and the organization's management. It must be achieved from the perspectives of the total system throughout the software and system development's lifecycles.*

Each of these triplet principles will be addressed in subsequent discussion, building on a variety of lifecycle software development methods, whose foci are cybersecurity, IA, and software reliability.

C. Risk and Vulnerability as the Byproduct of Software Development

The answer to the question, “What makes software at risk to cyber intrusion?” is indeed multifaceted and deserves a systems approach that addresses the entire lifecycle of software development from cradle to grave. *Risk is the result of a threat with adverse effects to a vulnerable system.* However, to measure risk, we defer to the definition by Lowrance [1976]: “Risk is a measure of the probability and severity of adverse effects.” Thus, to understand and measure risk, we must develop a systemic definition of vulnerability. We will do so through the centrality of state variables in the lifecycle of the software developmental process, and thus in cybersecurity and IA. For this purpose, it is constructive to define the following terms [Haimes 2004]:

- *Vulnerability* is the manifestation of the inherent states of a system (e.g., physical, technical, organizational, cultural) that can be exploited or otherwise adversely affected by terrorism, natural hazards, and accidents that result in harm or damage to that system. *Intent* is the desire or motivation of an adversary to attack a target and cause adverse effects.
- *Capability* is the ability and capacity to attack a target and cause adverse effects.
- *Threat* is the *intent* and *capability* to adversely affect (cause harm or damage to) the system by adversely changing its states.

Examples of vulnerabilities as manifestations of the inherent states of the system include the level of performance and system response, specific architectural design, developmental process, fidelity of telecommunications, system configuration, buffer overflows, and organizational structure [Chittister

and Haimes 2004]. Plakosh [2005] follows the spirit of the above definitions and defines software vulnerability as “A defect in the design and/or implementation of a piece of software that can be directly used by a third party to gain unauthorized access to a system or network.”

The premise that the vulnerability of a system is multifaceted, and is a manifestation of its states, implies that the ability to control the states of a system, a cybersystem for instance, would also control its vulnerability. Controlling vulnerabilities does not necessarily mean their elimination; rather, it makes the system more resilient and less susceptible to attacks. One major consideration for the efficacy of risk management in the context of information assurance and cyber infrastructure resilience and protection “is the ability to control the states of the system by improving its resilience. Primarily, this is the ability to recover the desired values of the states of a system that has been attacked, within an acceptable time period and at an acceptable cost.” [Haimes 2006]. For example, the lack of information assurance through cyber insecurity is intricately dependent on, and the byproduct of the vulnerabilities resulting from poor software development with unreliable commercial-off-the-shelves (COTS) products.

To be effective, risk management that is aimed at controlling the states of the systems must be followed throughout the software development lifecycle and by all the involved individuals. Examples of such states for SCADA and other cyberdependent systems, which were developed using Hierarchical Holographic Modeling (HHM), include [Haimes 2004; Chittister and Haimes [2004]: the system *configuration*; the *telecommunications* through which they are connected; their diverse *functionality* and the utilities they serve; their easy *access* by users; the *utility* of the SCADA system, e.g., electric power, oil and gas, water supply; the *temporal domain* during which they are acquired, designed, manufactured, tested, operated, manufactured, updated, and replaced throughout their lifecycles; the *management information systems (MIS)* with which they are controlled; and the *maintenance* of the system.

If we accept the premise that two factors—vulnerability and threat—lead to risk, and if we know that detecting and reducing the sources of threats are beyond the scope of this paper, then it is logical to ask: “What is the genesis of these software vulnerabilities? Are they controllable? If the answer is yes, how may they be controlled to yield a more resilient system?”

D. Resilience for Cybersecurity through Risk Management of the Software Lifecycle Developmental Process

To understand the complex process through which software is developed and integrated into a system, it is constructive to trace this process from all of its

critical dimensions and perspectives. Indeed, one may consider a hierarchy of such perspectives. For example, using Hierarchical Holographic Modeling (HHM) [Haimes 1981; 2004] to study the software development process from as many perspectives as possible, Higuera and Haimes [1996] identified several dimensions that affect the quality (meeting performance criteria), project-cost overrun, and time delay in meeting a schedule. Two major dimensions at the higher level of the HHM are here:

1. Temporal dimension:

- 1.1. Microvision: Specification; Solicitation (including request for proposals); Design and development (including architecture); System integration (including deployment and maintenance)

- 1.2. Macrovision: Conceptual design; Demonstration/validation; Engineering, manufacturing, development, and production; Maintenance and major update (including termination).

2. Human Dimension: Individual; Team; Management; Stakeholder (including customer and client)

Regarding the risk management of software development, there is a growing body of literature composed of a myriad of different approaches and methodologies [Haimes 2004]. For example, Sage [1992; 1995] presents risk from a lifecycle perspective, emphasizing the importance of considering the lifecycle of the entire project when implementing project risk management. He discusses the types of risks that can occur and the impacts they have on one another. Furthermore, he presents a set of tools and methods for managing risks to a project. Chapman [2001] emphasizes the importance of the risk identification process and advocates that a team-based approach is necessary to properly identify project risks. He presents several methods for eliciting sources of risk and discusses how the risk identification process affects the subsequent management of project risks. Two other methods are worthy of note. One is *continuous risk management* (CRM), where the entire set of risks is reexamined periodically to ensure that the set of critical risks is still a valid set. The other is *team risk management* (TRM), where the interactions among all parties involved in software development include a project's requirements and specifications, design and construction, finance and management, development of new technology, and response to a myriad of changes and conflicting signals from the many participating organizations (among others) [Haimes 2004]. Both CRM and TRM were developed by the Software Engineering Institute at Carnegie Mellon University [Dorofee et al. 1996]. These methods aim to engage the entire organization in the risk management process by continuously monitoring the risks to the project in order to manage them before they become major problems.

Adhering to the principles and guidelines of CRM and TRM, Nordean et al. [1997] adapted a holistic approach to risk assessment and management in

software development. The basic premise is that given a set of risks to a project and a set of strategies to manage them, it is necessary to track the status of critical systems to monitor the effectiveness of those strategies; thus, the need to identify meaningful risk metrics. To that end, working on the software development for the E-6 airplane, the Navy risk management team constructed one chart for each risk tracked and included each chart in a monthly risk report. This allowed the project management to quickly identify trouble spots throughout the lifecycle development of the software [Nordean et al. 1997; Pennock and Haines 2002].

To appreciate the role of systems integration in information assurance, and thus in countering cyberterrorism, it is instructive to review the evolutionary process that has been taking place in software engineering. This is the shift of system functionality and control from hardware to software, where the role of software engineering is increasing and is becoming more central in systems integration [Chittister and Haines 1996]. Furthermore, this shift has (i) displaced the responsibility of identifying sources of risk, (ii) introduced new sources of risk throughout the lifecycle of a system's development—its requirements, specifications, architectural design, process, testing, end product, and support systems for integration—and (iii) adversely affected the cybersecurity of the software-intensive system and its associated IA.

The fact that software increasingly has become the overall cross-functional system-integration agent has multiple implications for countering cyberterrorism, and for evaluating the changes that ought to take place in response to this shift. This overall pattern leads to two challenges. One is the need to reassess the educational requirements and the role of a new cadre of software systems engineers/systems integrators. Underlying this statement is the need to recognize software development as a true engineering discipline; thus, appropriate systems integration topics, which used to be (and remain) in the domain of hardware engineering, should be reintroduced into the software engineering curriculum. The other challenge is the need to assess the added vulnerability of software-intensive systems to risks of cyberterrorism, to develop appropriate risk management measures for countering such cyber risks, and to develop new and appropriate metrics with which to measure these risks.

Developing and institutionalizing the appropriate processes and practices as described in the Capability Maturity Model Integration (CMMI) [Chrissis et al. 2003] constitutes only one dimension of the technical and organizational complexity involved in software engineering development. *Integration, testing, and verification* are important functions for any product and are particularly critical for software. The *end product* originally conceived by the developer, however, may or may not represent a true realization of the actual customer's needs and desires. *It is in this lengthy and tedious developmental process that the*

need for information assurance is often overlooked, leading to significant adverse consequences to the systems which the developed software was intended to serve.

The thrust of the Task Force on Improving Security across the Software Development Process [2004] is to establish “a world with robust software security.” The authors of the report caution the professional community about the challenges associated with this thrust: “This is not an easy challenge and will take persistent, combined efforts of industry, academia, government and others to make long-term progress. Only by increasing security-oriented efforts throughout the software development lifecycle can we achieve this key component of the President’s National Strategy to secure Cyberspace.”

E. The Human Role in Information Assurance and Cybersecurity

The subject of this section is the human dimension—one of four major sources of risk associated with most systems; hardware, software, human, and organizational [Haimes [1991; 2004]. The human factor in risk is particularly critical to cybersecurity and information assurance because software development is a human-intensive activity. In addition, planning for and adhering to flawless systems integration throughout the lifecycle of software development also require a well-educated and well-trained cadre of software-systems engineers who master not only the art and science of software development, but also of systems and control engineering. This requisite is by and large provided by only a few institutions of higher education. In reality, the gap in the imperative need for this new cadre of software-systems graduate engineers is filled by on-the-job, home-grown professionals. Of course, there is nothing wrong with this on-the-job training; indeed, it is an urgent necessity. The problem is that only large and well-established public and private organizations can afford to provide such training, and hardly with the same rigor and uniformity of such programs offered through an established university curriculum.

What principles should guide such a software-systems engineering curriculum? The focal point of systems integration is the realization that all hardware-software systems are made of multiple interdependent subsystems. Each subsystem, in turn, is a system that is driven by its own state variables, inputs, outputs, and control and random variables, among others, where the output of one subsystem constitutes the input to others. Understanding this interconnectedness and the interdependencies among these many subsystems is imperative for an effective software architectural design and for ultimate systems integration and control. This fact is especially critical nowadays because, as mentioned earlier, software is increasingly assuming the roles of systems control and integration. Independent organizations such as the International Council on Systems Engineering (INCOSE), the Software Engineering Institute (SEI), the National

Laboratories, and other federally funded research and development centers (FFRDCs), may take the lead in requiring colleges and universities to enrich their curricula with courses on software-systems engineering. Such an initiative would be harmonious with the triplet principles advocated in this paper, and it would constitute a major step toward achieving information assurance and cybersecurity, given the non-diminishing risks of cyberterrorism in the foreseeable future.

The three principles of *Information Assurance and Cybersecurity through Software Management* call for adding new dimensions to the lifecycle model: requirements, specifications, architectural design, coding, integration, testing, and maintenance. The triplet principles advocated in this paper also highlight the centrality of quality and telecommunications fidelity, and the organization's culture, vision, and staff competence in achieving an acceptable level of security and information assurance. These and other attributes are addressed in the following sections.

F. The Educational Dimension

Ensuring the quality of software throughout its developmental lifecycle for the ultimate cybersecurity must address the technical educational knowledge and experience of the engineers engaged in this process. Three major engineering degree programs offered at colleges and universities are relevant to the subject of this paper: systems engineering, computer science/engineering, and electrical engineering. If the SCADA system resembles other engineering-based systems where the interface between hardware and software is achieved through systems integration and process control, then the expertise of engineers from all of the above three disciplines will be required (among others). It is unreasonable to expect that a four-year undergraduate degree in any of these three disciplines would produce engineers with expertise in all three majors, nor for that matter, would an additional two years of graduate education for a Master's degree. However, it is not unreasonable to expect that engineering students in any of the three majors should be able to understand and appreciate the contributions of others. For example, SCADA and other cyberdependent systems are composed of hardware (the common domain of electrical engineers), software (the common domain of software engineers), systems modeling and integration (the common domain of systems engineers), and process control and operations (the common domain of systems and electrical engineers). Consider the importance of process control and systems integration. The fact is that with the evolution of information technology (IT), software has assumed the role of process control; this presents a challenge to the above three engineering disciplines from the pre-IT education era. Although most systems engineering programs offer courses in software, only a few computer science/engineering programs offer basic courses in systems

integration and control (or require their students to enroll in them). Of course, the field of risk analysis, which is just as essential as the other majors for safety-critical systems such as SCADA and other cyberdependent systems, remains primarily the domain of graduate programs.

Another example is the difference in the focus of engineers between risk and reliability, the latter being the domain of study in systems engineering and electrical engineering programs. There are both historical and evolutionary reasons for the more common use of reliability analysis rather than risk analysis, as well as substantive and functional justifications. Historically, engineers have always been concerned with strength of materials, durability of product, safety, surety, and operability of various systems. The concept of risk as a quantitative measure of both the probability and consequences (or adverse effect) of a failure has evolved relatively recently. From the substantive-functional perspective, however, many engineers or decisionmakers cannot relate to the amalgamation of two diverse concepts with different units—probabilities and consequences—into one concept termed *risk*. Nor do they accept the metric with which risk is commonly measured. The common metric for risk—the expected value of adverse outcome—essentially commensurates events of low probability and high consequences with those of high probability and low consequences. In this sense, one may find basic philosophical justifications for engineers to avoid using the risk metric and instead work with reliability. Furthermore and most importantly, dealing with reliability does not require the engineer to make explicit tradeoffs between cost and the outcome resulting from product failure. Thus, design engineers isolate themselves from the social consequences that are byproducts of the tradeoffs between reliability and cost. The design of levees for flood protection is a case in point [Haimes 2004].

G. Triplet Principles of Information Assurance and Cybersecurity through Software Risk Management

Principle 1. Risk of software intrusion must be assessed and managed throughout the lifecycle of software development, focusing on both the functionality of software development and on the people involved in the process, knowing that hackers will exploit every weakness in the system.

At each step and phase in the software development lifecycle process we ask: “What can go wrong?” Hierarchical Holographic Modeling (HHM) [Haimes 1981, 2004] has been successfully deployed to answer this question. Although a detailed discussion on the use of HHM to identify the myriad sources of risk associated with SCADA systems and its use by railways has been presented by

Chittister and Haimes [2004], a brief description is presented here for completeness.

Hierarchical holographic modeling is a holistic philosophy/methodology aimed at capturing and representing the essence of the inherent diverse characteristics and attributes of a system—its multiple aspects, perspectives, facets, views, dimensions, and hierarchies. In a particular situation, a microscopic risk can become a critical factor in making things go wrong. In order to carry out a complete HHM analysis, the team that performs it must include individuals who have knowledge up and down the hierarchy.

HHM provides a mechanism for an organization to create an integrated risk assessment that covers all types of risk, ranging from low-level design issues to architectural issues to schedule issues to cost issues, among many others. Although each of the steps and phases in software development are important, here we choose the complexity of the architectural design as a vehicle with which to illustrate how HHM can be used for the realization of the above triplet principles of information assurance and cybersecurity through software risk management. By its nature, the architectural design involves myriad tradeoffs among such attributes as cost, speed, reliability, maintainability, security, and the overall desired performance of the software system, among others. The new principles of information assurance place security in a special position vis-à-vis all other attributes. Here, all tradeoffs made among all desired software attributes are tested against their impacts on the resulting level of information assurance; namely, security. In other words, similar to an acceptable product quality evolving in the 1980s from a three-sigma to a six-sigma measure of defects [Haimes 2004], so do the principles for information assurance mandate a paradigm shift in the fundamental tradeoffs made in the architectural design of software development. In general, for engineering-based hardware development there would be a finite and unambiguous number of fundamentally different paths or design options to meet a given set of design specifications [Chittister and Haimes 1993]. This is not so in the case of the development of architectural design of software; for any given specifications, the number of significantly distinguishable paths or design options for software is significantly larger, broader, and more ambiguous. This inherently greater freedom in design defies attempts to rely on historical statistics in predicting potential defects, faults, and errors in software development [Boehm 1981, 1988, 2006]. Because the design and development of engineering-based hardware products is markedly different from software development, it is substantially easier to prevent, detect, and correct defects, faults, and errors during hardware development. This paradigm shift from hardware to software architectural design is manifested by the plethora of the answers to the questions, “What can go wrong and how can the system be defeated?”

Bass et al. [2003] characterize cybersecurity “as a system providing non-repudiation, confidentiality, integrity, assurance, availability, and auditing.” They identify three major tactics for achieving cybersecurity, seemingly after the software has been already developed and installed in the systems. These are: resisting attacks, detecting attacks, and recovering from an attack. In contrast, in their discussion on the Architecture Tradeoff Analysis Method[®] (ATAM[®]), Kazman et al. [2000] maintain: “When evaluating an architecture using ATAM, the goal is to understand the consequences of architectural decisions with respect to the quality attribute requirements of the system...The architecture is the key to achieving—or failing to achieve—[the following] goals.”, In addition to raising architectural awareness and improving the level of architectural documentation, the goals of the ATAM are to record any risks, sensitivity points, and tradeoff points that are discovered when analyzing the architecture of the developed software.

Principle 2: *Achieving information assurance and cybersecurity must be placed high on the priority list of top management. {The two are intricately dependent on software quality and telecommunications fidelity}. This is synonymous with performing a holistic risk assessment and management.*

Principles and guidelines for what constitute the *quality* of a product or for software development are not a new phenomenon. For example, Garvin [1988] identifies the following eight dimensions or categories of quality as a framework for analysis: Performance, features, reliability, conformance, durability, serviceability, aesthetics, and perceived quality. Although Garvin addresses quality attributes for the production of hardware, the same principle is applicable for software development; namely, software quality is a multifaceted concept, particularly with respect to information assurance and cybersecurity.

Almost thirty years after Saltzer and Schroeder [1975] published eight secure-software development principles, Davis et al. [2004] maintained that they “apply just as well today as they did when they were first proposed in 1974:”

While the second principle embraces these and the other principles advocated by Humphrey [2004], it is driven by the premise that risk assessment and management must be an integral part of the overall decisionmaking process. This necessitates building on the principles and philosophy upon which systems engineering and risk analysis are grounded. In other words, at every stage of the software development process, the following triplet questions of risk assessment must be addressed [Kaplan and Garrick 1981]: “*What can go wrong? What is the likelihood that it would go wrong? What are the consequences?*” Similarly, the following triplet questions also must be addressed at every stage of the software

development process [Haimes 1991, 2004]: “*What can be done and what options are available? What are the associated tradeoffs in terms of all relevant costs, benefits, and risks? What are the impacts of current policy decisions on future options?*” A plethora of risk-based methodologies and modeling techniques have been developed to respond to the above two sets of questions [Haimes 2004], and thus, to support the second principle. Tradeoffs must continuously be made among all costs, benefits, and risks associated with the ultimate mission and functionality of the developed software, as well as with information assurance and cybersecurity.

To reflect on the inherent dependencies of the integrity and functionality of SCADA systems on IA and cybersecurity, and thus, on the fidelity of the supporting software engineering, consider the following attributes of these relationships. Myriad SCADA systems support a variety of infrastructure systems of different size and criticality, with disparate functionalities, across wide geographic regions, and through numerous telecommunications media that share uniform levels of security and safety. The hardware configurations and designs of these SCADA systems vary widely from one manufacturer to another in their fidelity, cost, complexity, and range of operations. Yet without exception, the operational integrity of all SCADA systems is dependent on the levels of IA and cybersecurity within which they operate and transmit their signals. What’s more, these SCADA systems do not operate in a vacuum; they are subject to the characteristics of the organizational professional environments within which they operate. Consider two organizations that use the same SCADA system and are being supported by the same telecommunications media. One has not adopted CMM and the other has achieved Level 5 (adopted and institutionalized the processes and practices at the top level of CMMI). Clearly, they cannot expect to enjoy the same level of IA and cybersecurity. Furthermore, the extent to which each organization develops and enforces security protocols and other measures, and whether their SCADA systems serve one or more headquarters, are also heavily dependent on their adoption of CMMI, and thus on the effectiveness of their “perimeter defense” and on “circling the wagons.” On the other hand, SCADA and other cyberdependent systems operated by software are likely to be less dependent on the effectiveness of the installed firewalls if their producers adhered to the three IA and cybersecurity principles throughout the software’s developmental lifecycle.

Principle 3: *Risk management of cyberterrorism must be the domain priority of the entire development team and the organization’s management. It must be achieved from the perspectives of the total system throughout the software and system development’s lifecycles.*

In their book *The Wisdom of Teams*, Katzenbach and Smith [1993], define what constitutes a successful team: “A team is a small number of people with complementary skills who are committed to a common purpose, performance and goals, an approach for which they hold themselves mutually accountable.” This last attribute—mutual accountability—is at the heart of the third principle; namely, managing risks of cyberterrorism must be the domain priority of the entire development team and the organization’s management, and be achieved from the perspectives of the total system. There is a growing body of literature on project risk management composed of numerous approaches and methodologies. Manufacturing firms, for example, commonly conduct a Failure Modes and Effects Analysis (FMEA) on the product and the assembly line, but may ignore the product development and design process [Haimes 2004]. Doing so may neglect addressing the risks inherent in defining the requirements and developing the software, by shifting the focus of the risk analysis process from the software development lifecycle to the software deployment phase. Sage [1992; 1995] presents risk from a lifecycle perspective, emphasizing the importance of considering the entire project lifecycle when implementing project risk management. Chapman [2001] emphasizes the importance of the risk identification process and advocates that a team-based approach is necessary to properly identify project risks. When developing software-intensive products and conducting the risk management process in teams, participants at all levels are imbued with a common purpose. While it is obviously not practical to have everyone in the participating organizations involved in all aspects of the project or on the risk management team, it is important to have a representative set, and an absence from team meetings does not necessarily imply absence from participation. An approach of total organizational involvement is likely to yield a more robust product than if risk management were simply tasked to a small group of risk experts [Haimes 2004].

Virtually any software engineering model or best practice, whether technical or product- or process-oriented, may be linked to risk management in one way or another. Some practices are explicitly about the risk management process and others are process or technology solutions that serve as risk mitigators. *Processes for Producing Secure Software: Summary of US National Cybersecurity Summit Subgroup Report* [Davis et.al. 2004] listed several software engineering processes and technical practices that impact security positively. In this vein, much of the SEI research agenda and technologies address risk management in one way or another. The SEI’s *continuous risk management* and *team risk management* [Dorofee et al. 1996] are in the category of risk identification and management practices, as is a specific process area of CMMI: Risk Management. *Continuous risk management* and *team risk management* aim to engage the entire organization in the risk management process by continuously

monitoring the risks to the project in order to manage them before they become major problems. The purpose of the CMMI Risk Management process area is to identify potential problems before they occur, so that risk-handling activities may be planned and invoked as needed across the life of the product or project to mitigate adverse impacts on the achievement of objectives.

In sum, adhering to these and other methods cited earlier provides a strong basis with which to realize the three principles of *Information Assurance and Cybersecurity Through Software Management*.

H. Summary and Conclusions

The information technology (IT) revolution and all the associated sectors of the economy that have become so dependent upon it are becoming increasingly at risk, in particular those critical infrastructures and sectors of the economy that are controlled through SCADA and other cyberdependent systems.. This is due to their inherent vulnerability to cyberattacks and other threats posed by various malevolent intruders. It is constructive to formally define vulnerability, threat, and risk, which are central to this paper [Haimes 2004; Haimes and Horowitz 2004]. *Vulnerability* is the manifestation of the inherent states of a system (e.g., physical, technical, organizational, cultural) that can be exploited or otherwise adversely affected by terrorism, natural hazards, and accidents that result in harm or damage to that system. *Intent* is the desire or motivation of an adversary to attack a target and cause adverse effects. *Capability* is the ability and capacity to attack a target and cause adverse effects. *Threat* is the *intent* and *capability* to adversely affect (cause harm or damage to) the system by adversely changing its states. *Risk* is the result of a threat with adverse effects to a vulnerable system.

Thus, the emergence of the myriad threats to cybersecurity in general and to information assurance in particular is a dominant risk to the beneficial uses accrued from IT. This has refocused attention on the lifecycle fundamentals in the development of software engineering. “Circling the wagons” through various firewalls after the development and installation of software into the system has proven to be a short-term remedy and ineffective in most cases, and also not cost-effective in the long term. The thesis advocated in this paper builds on the premise that IA and cybersecurity can be achieved through adhering to three principles.

These in turn build on several existing software engineering development methods that advocate quality, teamwork, and a commitment to appropriate organizational culture throughout the developmental lifecycle of the software. In particular, ensuring acceptable levels of reliability and integrity in telecommunications on which SCADA and other systems are so intricately

cyberdependent must be the domain of the entire lifecycle of software development, and not an aftermath patching process.

I. References

Bass, Len, Paul Clements, and Rick Kazman, *Software Architecture in Practice*, 2003, Addison-Wesley, Reading, MA, Second Edition.

Boehm, Barry.W., *Software Engineering Economics*, 1981, Englewood Cliffs, NJ: Prentice-Hall, Inc.

Boehm, Barry.W., "A spiral model for software development and enhancement." *Computer*, 1988, May, 61-72.

Boehm, Barry.W., "Some future trends and implications for systems and software engineering processes." *Systems Engineering*, 2006, **9**(1): 1-20.

Chapman, R.J., "The controlling influences on effective risk identification and assessment for construction design management." *International Journal of Project Management*, 2001 **19**(3): 147–160.

Chittister, Clyde, and Yacov Y. Haimes, "Risk associated with software development: a holistic framework for assessment and management." *IEEE Transactions on Systems, Man, and Cybernetics*, 1993, **23**(3): 710-723.

Chittister Clyde, and Y.Y. Haimes, "Assessment and management of software technical risk." *IEEE Transactions on Systems, Man, and Cybernetics*, 1994, **24**(2):187-202.

Chittister, Clyde and Yacov Y. Haimes , "Systems integration via software risk management." *IEEE-SMC Transactions*, 1996, **26**(5): 521-532.

Chittister, Clyde, and Yacov Y. Haimes, "Risks of terrorism to information technology and to critical interdependent infrastructures." *Journal of Homeland Security and Emergency Management*, 2004, **1**(4).

Chrissis, Mary Beth, Mike Konard, and Sandy Shrum, *CMMI: Guidelines for Process Integration and Product Improvement*. 2003, R Addison-Wesley, Reading, MA.

Computer Security Institute, *Cyber Attacks Continue, but Financial Losses Are Down*. rrichardson@cmp.com, Copyright 2003, Computer Security Institute, 600 Harrison Street, San Francisco, CA 94107. Telephone: 415-947-6320 Fax: 415-947-6023, email csi@cmp.com.

Davis, Noopur, and Julia Mullaney, "The Team Software Process (TSP) in Practice: A Summary of Recent Results," Technical Report CMU/SEI-2003-TR-014, and ESC-TR-2003-014, 2003 *Software Engineering Institute, Carnegie Mellon University*.

Davis, Noopur, Watts Humphrey, Samuel T. Redwine, Jr., Gerlinde Zibulski, and Gary McGrew, "Special report; processes for producing secure software: summary of US National Cybersecurity Summit subgroup report." 2004, *IEEE Computer Society*.

Dorofee, A. J., J. A. Walker, C. J. Alberts, R. P. Higuera, R. L. Murphy, and R. C. Williams, *Continuous Risk Management Guidebook*, 1996 Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

EIA, *Systems Engineering Capability Model*. (EIA/Is-731), 1998, Electronic Industries Alliance, Washington, DC.

Garvin, David A., *Managing Quality: The Strategic and Competitive Edge*, 1998. The Free Press, New York, NY.

Gilliam, David P., Thomas L. Wolfe, and Josef S. Sherif, "Software security checklist for the life cycle." *Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE '03)*, IEEE Computer Society.

Haimes, Y. Y., Hierarchical holographic modeling, *IEEE Transactions on Systems, Man, and Cybernetics*, 1981 **11**(9), 606-617.

Haimes, Y. Y., Total risk management. *Risk Analysis*, 1991, **11**(2), 169-171.

Haimes, Y. Y., and C. Chittister, A roadmap for quantifying the efficacy of risk management of information security and interdependent SCADA systems. *Journal of Homeland Security and Emergency Management*, 2005, **2**(2), 1-21.

Haimes, Y. Y., B.M. Horowitz, J.H. Lambert, J.R. Santos, C.Lian, and K.G. Crowther, Inoperability input-output model (IIM) for interdependent

- infrastructure sectors: theory and methodology., 2005, *ASCE Journal of Infrastructure Systems*. **11**(2), 67-79.
- Haimes, Y. Y., B.M. Horowitz, J.H. Lambert, J.R. Santos, K.G. Crowther, and C.Lian, Inoperability input-output model (IIM) for interdependent infrastructure sectors, II: case studies. *ASCE Journal of Infrastructure Systems*. 2005, **11**(2), 81-92.
- Haimes, Yacov Y., *Risk Modeling, Assessment, and Management*, 2004, John Wiley & Sons, New York, NY, Second Edition.
- Haimes, Y. Y., and P. Jiang, Leontief-based model of risk in complex interconnected infrastructures. *Journal of Infrastructure Systems*. 2001, **7**(1), 1-12.
- Haimes, Y. Y., and B. M. Horowitz, Modeling interdependent infrastructures for sustainable counterterrorism. *ASCE Journal of Infrastructure Systems*. 2004, 33-42.
- Higuera, R. P., and Y. Y. Haimes, Software risk management, Technical Report CMU/SEI-96-TR-012, ESC-TR-96-012, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, June 1996.
- Humphrey, Watts S., *Managing the Software Process*. 1989, Addison-Wesley Reading, MA.
- Humphrey, Watts S., *Winning with Software: An Executive Strategy*. 2002, Addison-Wesley, Reading, MA.
- Kaplan S., and B.J. Garrick, "On the quantitative definition of risk". *Risk Analysis* 1981, **1**(1), 11-27.
- Kazman, Rick, Mark Klein, and Paul Clements, *ATAM: Method for Architecture Evaluation*. 2000, CMU/SEI-2000-TR-004 and Esc-TR-2000-004, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.
- Katzenbach, Jon R. and Douglas K. Smith, *The Wisdom of Teams*. 1993, Harper Business, New York, NY.
- Lowrance, W. W., *Of acceptable risk*, William Kaufmann, Los Altos, Ca. 1976.

- McDermid, J.A. (Ed.), *Software Engineer's Reference Book*. 1991, Oxford, England: Butterworth.
- Nordean, D. L., R. L. Murphy, R. P. Higuera, and Yacov Y. Haimes, *Risk Management in Accordance with DODD 5000 Series: An Executive Perspective*. 1997, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.
- Pennock, M. J., and Y. Y. Haimes, "Principles and guidelines for project risk management." *Systems Engineering*. 2002, **5**(2): 98–108, 2002.
- Plakosh, Daniel, "Coding flows that leads to security failures." 2nd Annual Hampton University Information Assurance Symposium: *Building Information Assurance Capacity and Improving Infrastructure at Minority Serving Institution*, April 2, 2005. Hampton University.
- Sage, A., *Systems Engineering*. 1992, John Wiley & Sons. New York, NY
- Sage, A., *Software Systems Engineering*. 1995, John Wiley & Sons, New York, NY.
- Saltzer, Jerome and Michael Schroeder, "The protection of information in computer systems." *Proceedings IEEE*, 1975 **63**(9) 1278-1308.
- Ron Moritz and Scott Charney, Co-chairs, Task Force on Improving Security across the Software Development Lifecycle, Microsoft, April 1, 2004.
- Wulf, William A. and Anita Jones, "A perspective on cybersecurity research in the United States." In *Terrorism: Reducing Vulnerabilities and Improving Response*, 2004, National Research Council of the National Academies, Washington, DC.