What’s Inside

From the President 3
Critical System Behaviors of the Future 3

Special Feature 6
The Buck Stops Here: Systems Engineering Is Responsible for System Security 6
Management Initiatives to Integrate Systems and Security Engineering 10
System Security—Shaping or Impeding Systems in the Future? 13
What Does a Systems Security Engineer Do and Why Do Systems Engineers Care? 16
Addressing Attack Vectors Within the Acquisition Supply Chain and the System-Development Lifecycle 19
Requirements Challenges in Addressing Malicious Supply Chain Threats 23
Sustainable Agile Security Enabled by Systems Engineering Architecture 30
Security Engineering Models 34
System Integration at the Security Interfaces 41
Verifying Security-Control Requirements and Validating their Effectiveness 45
An Approach to Integrate Security into a Systems Engineering Curriculum 49

Forum 55
Affordable Requirements Verification 55

Technical Operations 58
Technical Directions: Camping Out 58
A Library of Systems Engineering Case Studies: Making the Case for Systems Engineering 59
Report on the 2013 Workshop of INCOSE’s Systems Engineering and Architecting Doctoral Student Network 60

INCOSE Spotlight 64
INCOSE Spotlight on . . . Wolter J. Fabrycky 64

INCOSE Foundation 65
Valerie Gundrum Engineering Scholarship Fund 65
Nominations Now Open for the David Wright INCOSE Leadership Award 65

Book Reviews 66

Final Thoughts From the Chief Editor 71
Program Protection Planning
Online Graduate Certificate

Worcester Polytechnic Institute (WPI) is the first to offer a Graduate Certificate in Program Protection Planning—specifically designed to meet the demands put on today’s systems engineers and defense industry. Our program is six courses (18 credits) and **available online**. It can also be delivered at your organization and customized to meet specific needs and goals.

In response to the Department of Defense’s strong emphasis on reliable plans, our subject matter experts have outlined essential topics organizations need to remain, and become more, successful:

- Concepts of Systems Engineering
- Dependable Systems and Networks
- Supply Chain Risk Management
- Protection Planning Across the Program Life Cycle
- Practical Applications of Systems Security Engineering
- Software Security Design and Analysis
- Operations Risk Management
- Network Security

Go to [cpe.wpi.edu/+ProgramProtection](http://cpe.wpi.edu/+ProgramProtection) to discover more about this online graduate program.
**Critical System Behaviors of the Future**

In a past “President’s Corner” (INSIGHT vol. 16, no. 1, pp. 3–4), I shared with you my views concerning the value of a systems engineer. I crafted the discussion as if I were speaking to the directors, vice presidents, and general managers of our organizations. These titles are influential to the success of our careers and in determining the level of our salaries. In essence I said that the job of the systems engineer is to deliver the behaviors of a system that provide stakeholder value, and by delivering value, to provide sales and profit to our organizations. In the past, performance was the king of system behaviors. While performance will always be important, I believe we are in a world where other system behaviors become as critical as performance for effective market penetration and differentiation from our competitors.

To deliver system behavior, the systems engineer must define a group of subsystems and precisely how those subsystems are to interact with each other. It is the subsystems and their interactions which produce the system-level behavior.

Many of us recognize a vehicle that can take a 60-degree curve at 200 miles per hour as possessing a valuable system behavior. Would we as quickly recognize behaviors as attributes of the parts of our system but not the system as a whole, then we are likely to consider them as jobs for the “specialty engineers.” I’ve looked back into past behaviors of our system engineering community. What I find are examples of systems engineers giving our “specialty engineering” colleagues these challenges by way of the requirements-allocation process. I think we have been wrong to do this. Our “specialty” colleagues are likely to take these allocated requirements and focus on building safe, private, trusted, available parts of a system—rather than in delivering safe, private, trusted, and available system behaviors. It is true you can build a safer system by building safe parts. However, you can’t build a truly safe system without having safe parts interacting with each other in a safe manner. The same can be said for other system behaviors (private, trusted, available, and so on).

To do our jobs as system engineers, we must have “specialty systems engineers” tightly integrated into the team. Our relationships, both organizationally (INCOSE) and individually need to be strengthened across the “specialty” organizations and with our “specialty” engineering colleagues. Tight collaboration will result in approaches that grow and reward “specialty system engineers” within both our community and the “specialty” communities.

I believe the organization whose systems engineers consider the delivery of these critical system behaviors as part of their job will prosper and survive swings in the market. The time is near where a stakeholder will likely choose the highly available, safe, protected, trusted vehicle that takes a 52-degree curve at only 180 miles per hour, over the vehicle that achieves the desired 60-degree curve at 200 miles per hour. In my travels, our stakeholders are telling me they need help with the delivery of these critical system behaviors. They look to INCOSE to help them find the right path and approach. INCOSE (and I) look to you, our members, for the thought leadership to guide us down that path.
Johns Hopkins Engineering
Online and On-Site, Part-Time Graduate Programs

Systems Engineering
MS in Systems Engineering
MSE in Systems Engineering

AVAILABLE FULLY ONLINE
7 FOCUS AREAS
2,600+
TOTAL COURSE ENROLLMENTS FOR 2012–2013
LEARN MORE. APPLY TODAY.
EP.JHU.EDU/INCOSE
1.800.548.3647
Corporate Advisory Board — Member Companies

Air Force Center for Systems Engineering
Airservices Australia
Alliant Techsystems
Analytic Services-Applied Systems Thinking Institute
Astrium, an EADS Company
ATKINS
Aviation Industry Corporation of China
BAE Systems
Bechtel
Beihang University School of Reliability and Systems Engineering
Boeing Commercial Airplane Co.
Boeing Defense, Space & Security
Boeing Research and Advanced Technology
Booz Allen Hamilton Inc.
C.S. Draper Laboratory, Inc.
Carnegie Mellon University Software Engineering Institute
Cassidian
Cranfield University
Cummins, Inc.
Defense Acquisition University
Deloitte
Deputy Assistant Secretary of Defense for Systems Engineering, US Department of Defense
DRS Technologies, Inc.
EADS N.V.
Exelis
Federal Aviation Administration US
Ford Motor Company
General Dynamics
General Electric
George Mason University
Georgia Institute of Technology
Honeywell International
IBM Corporation
JAXA (Japan Aerospace Exploration Agency)
Jet Propulsion Laboratory
Johns Hopkins University
Keio University
L-3 Communications
Lockheed Martin Corporation
Los Alamos National Laboratory
ManTech International Corporation
MAP systeme
Massachusetts Institute of Technology
Medtronic, Inc.
Missouri University of Science and Technology
National Geospatial–Intelligence Agency
National Reconnaissance Office
National University of Singapore
Naval Surface Warfare Center—Dahlgren Division
Northrop Grumman Corporation
PA Consulting
Pacific Northwest National Laboratory
Proctor & Gamble
Raytheon Corporation
Rockwell Collins, Inc.
Rolls-Royce
Saab AB
SAIC
Sandia National Laboratories
Scitor Corporation
SELEX ES, a Finmeccanica Company
Siemens
Singapore University of Technology and Design
SRA International
Stevens Institute of Technology
Swedish Defence Materiel Administration
TASC, Inc.
Thales
The Aerospace Corporation
The MITRE Corporation
The SI
The University of New South Wales, Canberra
TNO Technical Sciences
UK MoD
United Technologies Corporation
University of Maryland
University of South Australia
Defense and Systems Institute
University of Southern California
University of Texas at El Paso
US Army ARDEC
US Army TARDEC
Vitec Corporation
Volvo Construction Equipment
Woodward Inc
Worcester Polytechnic Institute
Who is responsible for systems security? As shown in figure 1, the acquirer (Acq) thinks it is the supplier, the supplier (Sup) delegates that responsibility to systems engineering, who pass it on to system security engineering (SSE), which meets requirements originating with the acquirer. This arrangement results in a finger-pointing circle when security fails.

New revisions to the INCOSE Systems Engineering Handbook are integrating responsibility for system security into the systems engineering processes. Placing responsibility on systems engineering is only a first step. A second step requires mutual engagement between systems engineering and security engineering, an engagement that can only be enabled by systems engineering. Systems engineers and program or project managers will be expected to engage effectively throughout the systems engineering processes and activities—beginning with requirements analysis and the concept of operations, and proceeding through the full lifecycle of development, operations, and disposal.

The theme articles in this issue of INSIGHT focus on the nature and problems of effective security engineering engagement in critical systems engineering processes. In the end, the acquirer and the supplier must also engage, in a shared responsibility that recognizes and deals with an unpredictable future of security threats. But that is another story: we that cannot be effective until systems and security engineering engagement is achieved.

As systems engineers we find that our systems are under attack by intelligent and innovative adversaries. We rely on the specialty engineering function of system security engineering to protect what we design, build, and deploy. Unfortunately the results are not encouraging. The costs invested in system security engineering and deployed security technology are increasing every year, while the losses caused by breaches are also increasing every year. Something is not right. Something needs to be done differently.

In government acquisition projects, systems engineering security concerns are driven by security requirements specified by the customer, generally in the form of adherence to policy and standards. In commercial product development, systems engineering security concerns are driven by market acceptance, measured in continued product-line revenues and competitive position. Hybrid projects include commercial acquisitions of one-off systems (such as custom banking systems) and government projects of repetitive system deployments (like Predator drones).

Security technology has relied principally on reverse engineering successful breaches after the fact, and developing ways to recognize and prevent those attempts when tried again. This is true in the evolution of cyber security with firewalls and intrusion detection mechanisms, in physical security with various intrusion sensors and anti-tamper devices, in human security with security-clearance methodologies and employee behavior profiling, and now moving into supply-chain security with trusted supplier qualification and commercial off-the-shelf (COTS) device and software security testing.

As systems engineers we rely on the systems-security-engineering profession to know the methods of security breaches and to know the methods for detection and prevention.
The Logic of Systems Engineering Responsibility for Systems Security

Systems are engineered with expectations: to provide services or carry out missions that justify investing in development, production, and sustainment. The return on investment occurs over a time period of many years. Value fails to accrue if the system’s life or its ability to carry out its mission during that life is less than required. System lifetime, protection of critical system information, and critical assets that may be protected by a system, are under threat by competitive entities, as well as by unanticipated situations. System security is the property that guards against and counters these threats—a purposefully engineered property that can only emerge successfully from thoughtful system engineering.

Emerging technology is a double-edged sword. Modern technology is both the enabler of remarkable system capability and a source of constantly-evolving adversarial attack. The increasing use, knowledge, and complexity of digital data, control systems, and communication networks compel both new system capability and new vectors for system compromise. Accessibility to technologies such as global positioning systems, drones, and biological intervention bring new capability to physical system intervention. Globalization and outsourcing have made supply-chain insertion a successful new vector of system intervention. Moreover, enduring human factors of selfish interests, ideological motivation, and occasional faculty impairment make the insider threat always likely and multidimensional.

Within the systems engineering taxonomy, security is classified as a specialty engineering activity. To be sure, special knowledge, experience, and practice are necessary in system security engineering; especially when systems of all kinds are targets for intelligent, resourceful adversaries intent on system compromise. Security engineering is engaged to make a system secure, but when allocated solely to a separate specialty activity, this engagement is constrained by the nature of an already defined and often implemented system, or limited to ensuring that called-for standards and regulations are met. Constrained evolution of existing systems, and characterization as a compliance activity, hamstring the ability of security engineering to accept and dispatch system security responsibility effectively.

Dispatching Responsibility

Systems engineering is described and practiced as a collection of technical and project processes, organized for disciplined execution, with checks and balances throughout—in prudent practice. At the highest level the technical process of verification and validation, with test and evaluation, is focused on verifying that the system meets requirements and that the requirements are valid for meeting the system intent. As outlined in the INCOSE Systems Engineering Handbook, within each of the system engineering processes there are also formal internal checks and balances, called out to ensure the integrity of each process discipline.

Verifying and validating sustainable security of a system reaches back to the earliest two system engineering processes of defining stakeholder requirements and analyzing requirements, where requirements and the concept of operations govern what will be verified and validated for system security. Important outputs of the requirements analysis relevant for system security include measures of performance, systems functions, and verification criteria. Systems functionality should not ignore those functions that are intended to provide sustainable system security, nor can dedicated system security functions preclude the need for all other functions to include appropriate internally-integrated security measures. The expertise for integrating sustainable security in the processes of stakeholder-requirements definition and requirements analysis is best provided by the specialty engineering resources of security engineering as full peers, enabling the rapid upgrade and augmentation of security measures.

The concept of operations should recognize the reality of an evolving and innovative threat environment. This recognition should influence system-architecture considerations that will facilitate sustainable system security measures, so that these measures can evolve continuously throughout development and throughout operational life.

System architecture enables or impedes system security, and is an early design activity where engagement of security engineering is important. System adversaries learn system-protective measures and change methods rapidly. Architecture must accommodate protective measures that can change just as rapidly, and resilience that can deliver functionality while under attack. These needs argue for a security architecture that is composed of loosely coupled encapsulated functional components that can be replaced, augmented with additional functionality, and reconfigured for different interconnections. Long system life expectancies are critically vulnerable to non-agile architectures.

In each of the system engineering technical processes, disciplined checks and balances are included to ensure process integrity. Each of these processes enables or constrains the end capability of sustainable system security, and thus warrants explicit attention and collaboration with the expertise of actively engaged security engineering resources.

Trade-off evaluation and decision are important functions of system engineering, but these evaluations and decisions should be informed and advised by the expertise of competent and thoughtful security-engineering resources. Competence is rooted in the depth of specialty knowledge, whereas thoughtfulness is enabled by the breadth of the full system’s requirements and intent knowledge—which can only be obtained when security engineering is in full participation throughout all of the systems engineering processes.
Experience Speaks

The articles in this issue of INSIGHT are intended to help lower and remove the barriers to mutually effective engagement of systems and security engineers. The barriers are those perceived by systems engineers, security engineers, project managers, and program managers. Many of these articles provide experience examples that can help systems engineering accept and dispatch responsibility for the sustainable security of systems. Systems engineers must recognize that systems security cannot be effective if it is not integrated intimately with the system requirements, the concept of operations, the architecture, and all the other systems engineering processes through operation and disposal.

Management Initiatives to Integrate Systems and Security Engineering

From Raytheon, Lori Masso and Beth Wilson share a management initiative that places system security responsibility within the systems engineering processes. This responsibility is backed up with system engineering training that provides fundamental understanding of system security concepts and policies and addresses how to identify security requirements. It also provides enough knowledge of the security fields to be able to ask the right questions and know if the answer represents a reasonable approach. Lori is a Principal Systems Engineer at Raytheon Company, with seven years of experience in system security engineering. Beth Wilson is an INCOSE ESEP, INCOSE Systems Security Working Group cochair, a Principal Engineering Fellow at Raytheon Company, and US National Defense Industrial Association Development Test and Evaluation Committee cochair, with a PhD in electrical engineering from the University of Rhode Island (US).

Information Security: Shaping or Impeding Systems in the Future?

Ken Kepchar, ESEP, retired Chief Systems Engineer of the US Federal Aviation Administration in the NextGen office, and owner of EagleView Associates, offers systems engineering consulting and training with a focus on transportation-related issues. Ken’s article raises concern over the landscape of shifting digital technologies that influence systems engineering decision making. He notes that new risks are being introduced while traditional system development efforts defer or ignore security considerations until after the functional architecture has been established. He outlines some commonly held security “myths” that need to be purged, some principles to employ for effective security integration, and adjustments to include security capabilities as contributing feature in system design.

What Does a Systems-Security Engineer Do and Why Do Systems Engineers Care?

Janet Oren suggests that integration of systems-security engineering with all systems engineering processes is on the cusp of achievement. She attributes this to growing expertise in the security engineering community, and to a more detailed process approach expected in 2013 from the US National Institute of Standards and Technology as Special Publication 800-160, Systems Security Engineering. Janet is a Technical Director and Systems Security Engineer for the US Department of Defense, with a PhD in systems engineering from Stevens Institute of Technology (Hoboken, US-NJ). She feels that the success of this integration will result in systems that protect information and are more resilient.

Addressing Attack Vectors Within the Acquisition Supply Chain and the Development Lifecycle

John Miller of The MITRE Corporation opens a discussion of supply-chain threat. From a systems engineering view he focuses on understanding and addressing the “attack vectors” used to exploit vulnerabilities in the system-acquisition supply chain and the system-development lifecycle, examining the intersection of attack vectors with activities of systems engineering. John is a systems engineer at MITRE with expertise in system security engineering, software engineering and development, hardware–software integration, and project management. He is currently developing program-protection methodologies and frameworks for the US defense department’s major acquisition programs.

Requirements Challenges in Addressing Malicious Supply-Chain Threats

Paul Popick and Melinda Reed continue the discussion of supply-chain threats with latest US Department of Defense state of practice for incorporating trusted system and network security requirements into the specifications for large, complex systems. They describe the current environment, the trends that are influencing the need for system security engineering, and the types of system security requirements and analysis techniques. Paul is a retired IBM Global Services Director of delivery excellence is cochair of the INCOSE System Security Engineering Working Group, and maintains a continuing interest in systems engineering and program management through teaching and consulting. Melinda Reed is the Deputy Director for Program Protection within the Deputy Assistant Secretary of Defense Systems Engineering organization of the office of the US Secretary of Defense.


From Sandia National Laboratories (Albuquerque, New Mexico, US), Ruth Duggan and Mark Snell address the complicating factors in developing system security requirements. They suggest that an expert system security engineer can help the systems engineer navigate these complications so that the resulting system will be robust against future threats and technical advances. Ruth is a Senior Member for the
Enabling Sustainable Agile Security Through Systems Engineering

Rick Dove notes that long-life systems will have functional upgrades and component replacements throughout their life. Continuous evolution of system security is necessary to maintain parity with a continuously evolving threat environment. He reviews agile architecture fundamentals that enable effective security evolution, the important role played by the concepts of operations, principles for fleshing out the architecture, and a framework for developing responsive requirements. Rick teaches agile and self-organizing systems at Stevens Institute of Technology, chairs the INCOSE working groups for Systems Security Engineering and for Agile Systems and Systems Engineering, and is CEO and principle investigator for security-technology contracts at Paradigm Shift International.

Security Engineering Models

From Sotera Defense Solutions, Bob Marchant integrates the systems engineering lifecycle model with the US National Institute of Standards and Technology Risk Management Framework used as a security engineering lifecycle model. He then walks through the activities and guidelines used in process models and system baseline models that structure the systems security engineering effort. Bob is a CISSP (Certified Information Systems Security Professional) and an ISSEP (Information Systems Security Engineering Professional), and a technical fellow at Sotera, with 35 years of systems engineering experience that includes 20 years involved with information-systems security.

Evaluation of Security Risks using Mission Threads

From the Software Engineering Institute, Carol Woody describes the use of mission thread security analysis as a tool for systems engineers to evaluate the sufficiency of software security requirements. She then shows the value and use of this approach with a detailed example system of the Commercial Mobile Alert System, a system that disseminates geographically targeted emergency-alert notifications. Dr. Woody leads a research team at the Software Engineering Institute focused on cyber security engineering: building capabilities in defining, acquiring, developing, measuring, managing, and sustaining secure software for highly complex, networked, software-intensive systems.

System Integration at the Security Interfaces

Kevin Stoffell, a Cyber Security Architect with the Battelle Memorial Institute, notes that security in information-technology systems is typically distributed, with many components relying on other components to ensure some or all of their security. Kevin suggests that this distributed interdependency poses some problems with integration process. He provides an example to illuminate the nature of the problems, and suggests that systems engineering interface control documents can and should be used as support to overcome these problems in the system certification, accreditation, and authorization processes.

Verifying Security Control Requirements and Validating their Effectiveness

From Thales Australia, Bruce Hunter illuminates the planning and methods for system security verification and validation, and addresses continued verification and validation throughout a system’s operational lifetime. He stresses the need for an adaptable approach that accommodates emerging or discovered threats and vulnerabilities. Notably, he advises setting the scope of security testing beyond the identified system security requirements, to include any path that a threat may exploit. Bruce works in quality, security, and safety assurance for Thales, and holds CISM (Certified Information Security Manager) and CISA (Certified Information Systems Auditor) credentials from the Information Systems Audit and Control Association.

An Approach to Integrate Security into a Systems Engineering Curriculum

Don Gelosh wraps up our theme by addressing the education of systems engineers, proposing that system security consciousness and knowledge be integrated throughout the curriculum, especially in courses that deal with requirements, architecture and design, risk management, integration and test, sustainability, scalability, and flexibility. He provides a framework for consideration and tailoring by institutions offering degrees in systems engineering, and speaks from personal experience and responsibility. Don is a CSEP-Acq, the first Director of Systems Engineering Programs at Worcester Polytechnic Institute, and is Level III qualified in the US Department of Defense systems planning, research, development, and engineering career field.

A Closing Thought

We cannot put security and system sustainability (an ility in name only?) into the functional category, as that category has historical meaning that refers directly to system functional requirements of delivered features. But that seems fuzzy. Security will not have the priority it needs until it is recognized as a functional requirement. Note that an insecure system is inherently “non-functional.” Is this all a semantic game, or is it a game of I-Don’t-Want-To-Have-To-Think-About-That? 😊
Management Initiatives to Integrate Systems and Security Engineering

Lori Masso, lori_a_masso@raytheon.com; and Beth Wilson, elizabeth.wilson@incose.org

SYSTEMS engineers are responsible for all aspects that comprise a successful system, including an understanding of security requirements and design aspects to mitigate security risks. Security requirements are typically specified at a system level and address a variety of capabilities. These capabilities include information assurance, situational awareness with persistent monitoring, supplier integrity, anti-tamper, information operations, identity-access control, authentication, and cryptography. These requirements will not be satisfied by a simple checklist, but rather these capabilities must be embedded into the system architecture and design in order to ensure their successful and effective implementation.

Company Initiatives

While individual systems engineers and systems-security experts understand the need for embedded solutions, Raytheon Company modified its systems engineering processes to ensure that system security engineering is an integrated part of our system-design activities. In other words, Raytheon Company leadership decided that system security engineering is a systems engineering responsibility. Leadership addressed this through top-down initiatives in the areas of systems engineering organizations, process initiatives, and training. These initiatives included identification of system security engineering roles, responsibilities, and activities, as well as new process enablers and training programs across the organization. The resulting changes are endorsed by the discipline councils comprised of the engineering directors across the company.

The process changes and enablers are largely based on the US Department of Defense Program Protection Plan for the “what” activities and definitions. The real value in the enablers, however, is the definition of “who” and “how” to successfully plan programs and implement designs that result in secure systems being delivered. In some parts of the company, this first initiative resulted in an expansion of the systems engineering organization to include a department comprised solely of security engineering subject-matter experts that are shared across the business. These individuals are also referred to as cyber practitioners. This organization ensures horizontal protection consistency across programs, services, and the enterprise.

Next, leadership identified the need to assess system requirements, including system security requirements, during the proposal phase of a program, and in some cases, prior to receipt of a request for proposal. The Statement of Work and Contract Deliverables are reviewed to ensure the security requirements and contractor responsibilities are understood and the system security scope is adequately addressed in the proposal cost and technical volumes.

Finally, leadership recognized the need to describe roles and responsibilities for all disciplines, and identify process owner, stakeholders, and approval authority. The systems engineering process was modified to align with and embed security engineering design and development into system design activities. The leadership endorsement ensures that compliance with defined activities and success criteria are reviewed and confirmed by independent reviewers prior to authorization to proceed to the next stage of the program. For example, this is accomplished through Independent Review checklists and Integrated Product Development Gate Reviews (e.g., system functional requirements review, system and software critical design reviews).

Security solutions provide an ability to monitor and assess the indications of attacks or unauthorized instrumentation, and systems engineers collaborate with their customers with respect to interpretation of tamper indications. Many security solutions provide little or no data output with which to evaluate the performance of the system or security performance. This is a particularly important attribute for anti-tamper solutions. Raytheon’s process enablers identify security goals and objectives with sample material. For the objectives identified early in the program (ideally during the proposal), security metrics are established with data collection and reporting details.
The resulting process tools are mapped to their purpose and benefit during system design, and they aid in identifying functional areas that are often significantly impacted by system security requirements. These areas include, but are not limited to system integration and testing, specialty engineering (i.e., safety, reliability, availability, maintainability and human factors), operator training, and integrated supply chain. The next section will describe these focus areas in more detail.

**System Engineering Focus Areas**

The system engineering process begins with trades associated with minimizing risks and continues to update these risk assessments throughout the program and system lifecycle. System security risks are an important part of this ongoing risk assessment and trade study effort. As a system design evolves and the system-level security requirements are decomposed into component-level requirements, the impacts on other aspects of the system are evaluated. In particular, security requirements may significantly impact system integration and test, operational availability assessments, maintenance and logistics, and operator training. Recurring operations and sustainment costs are typically much greater than the non-recurring development costs, and system engineering design process includes an assessment of how a security solution will impact these phases of the product’s lifecycle.

**System Integration and Test**

A successful system design will balance need to know, need to share, and need to use. System access based on need-to-know permits valid users to interact with the system as designed and prevents unauthorized access. Typically, security solutions for access control are more obvious to the operator, such as information assurance implementations like user authentication. However, need-to-use is an operational consideration of significant importance. System engineers and system security engineers will agree on the complexity of the access control (for example, is the operator required to memorize lengthy passwords and carry a token generator with them?). Access controls must be implemented such that they do not interfere with time-critical operations. Preventing valid users from accessing a system can be irritating to a customer and potentially deadly to a war-fighter. System performance may also depend on access to data from other systems, so need-to-share must provide timely data to integrated systems while preventing unauthorized access or malicious modification of shared data.

System integration and test often relies on instrumentation, data collection, and analysis. While these capabilities may be allowed in a secure test facility, they are problematic on production hardware in a fielded location or when a system is integrated into a system of systems. The systems engineering challenge is to engage the security engineers early in the integration and test planning process in order to understand the limitations that a security solution may impose on integration and test. Creative collaboration between systems engineers and system security engineers is encouraged in order to develop effective integration and test strategies and approaches. For example, the security implementation could be configured in varying stages of “lock down” during system integration and test. Early in development, the security implementation may allow debug and data collection for evaluation of security performance. As the system capability matures, the data collection may be reduced to a minimum set that is necessary to evaluate the overall observable system performance but not the security capabilities. Regular collaboration between system engineers and system security subject matter experts, as well as the appropriate government or third-party evaluation agencies, will reduce integration and test risk, and will likely result in the early discovery and correction of potential testing deficiencies.

**Integrated Supply Chain**

During the design and development of hardware and software, the systems engineering design process includes an understanding and alignment with the integrated supply chain team to adhere to the processes and policies in place to ensure parts, materials, and supplier integrity. These policies are in addition to obsolescence management. Process enablers address third-party suppliers, third-party sources, subcontractors, and commercial off-the-shelf vendors. Parts selection, particularly for commercial off-the-shelf hardware and software components, ensures the integrity of the supplier, as well as the availability and source of the part for the lifetime of the system with a significant focus on mitigation of counterfeit parts. In some cases, a lifetime buy is the recommended solution and must be identified early in the program’s lifecycle (that is, during preliminary design) so that the cost is known in advance.

**Specialty Engineering**

As system-level security requirements are decomposed into component requirements, the system engineering process includes engagement and coordination with system security engineers and specialty engineers to focus on safety, reliability, availability, maintainability, and human factors. System security engineers will provide solutions that are designed to react under certain conditions (such as cyber attack), and these reactions may include disabling hardware, which must be assessed by safety engineers. The system design process includes considerations for whether these conditions could occur during normal operations, which may
induce a false event and reduce operational availability and impact mean time to repair and mean time between essential hardware failures. If the system invokes an observable reaction when an attack is detected, the system engineer and system security engineer will agree on how to notify the operator or maintainer in the field. For example, will a mission-critical line-replaceable unit fail, terminating a mission? Or will the operator or maintainer no longer be allowed to successfully log into the system? Or will the system performance degrade in an observable way?

Once this is decided, the operator or maintainer will be provided an indication (such as displaying a failed component that needs to be replaced) and understand how to recover in a reasonable and predictable amount of time. This imposes human-factors requirements on interfaces to the operator or maintainer to provide clear indicators of system health and status, as well as the logisticians to ensure the availability of the software or hardware required to recover system operations. System-level availability requirements may dictate that spare hardware and software must be stored near a fielded system. Depending on the specific location and sensitivity of the system and its components, additional requirements may be imposed on the storage facility to address aspects such as ownership, access control, physical size, and environment.

So who are the operators and maintainers for the system once it is fielded? The level of exposure during maintenance activities will be identified early in the system design because it will drive the system security solution. This exposure is typically captured in the documentation of the system design and architecture, and the systems engineers and system security engineers will partner to define an approach with their customers to mitigate maintenance exposures and potential compromises as early as the system requirements review.

How are the operators and maintainers trained? An effective training program that focuses on maximizing system performance is essential. However, the training methods will be evaluated to ensure that they do not expose the security solution. The system-design process will include the development of training programs and tools in order to properly address potential exposures. A comprehensive training program will provide an operator and maintainer with sufficient information to operate the system (without dependence upon complicated instructions), detect and isolate failures, and conduct both preventative and corrective maintenance actions without introducing security risks. For example, some training programs employ simulations that utilize tactical or, in some cases, development hardware and software in a standalone environment. While these simulations provide a realistic training environment, the same hardware and software is integrated with a security solution in the real system. System security engineers will understand who provides the training, who receives the training, how the training material is documented and controlled, and what types of software capabilities and hardware tools or equipment are required. A successful training program will depend on the system engineers and system security engineers to design a system that provides diagnostics, fault-detection, and fault-isolation capability without introducing exposures to the security solution.

System Engineering Training

Given the alignment and coordination required between systems engineers and system security engineers throughout the system’s lifecycle, leadership identified the need for system engineering training in order to design systems with effective, embedded security solutions. This systems engineering training provides fundamental understanding of system security concepts and policies and addresses how to identify security requirements. It also provides enough knowledge of the security fields to be able to ask the right questions and know if the answer represents a reasonable approach. A reasonable approach balances the design compliance to security requirements against impacts to program cost, schedule, technical risk, system performance, operations, and sustainment. Therefore, systems engineering training has been updated to expand the focus on systems security.

Security technologies and techniques are not restricted to the experts, although there may be limitations on what can be taught in a classroom. Raytheon has implemented a range of such training internally. Systems engineers can enroll in online awareness modules available on demand to reinforce common approaches and vocabulary. Raytheon also offers more detailed classes for cyber practitioners, as well as “boot camps” that combine classroom learning and skills development via experiential learning in an intensive program for selected practitioners. Leadership recognizes that training is needed on a full spectrum of awareness and expertise to increase proficiency in systems security across the systems engineering workforce.

Summary

To successfully integrate systems security engineering into systems engineering, Raytheon recognizes that it must change the way systems engineering processes are implemented. Yesterday’s solutions will not work for today’s problems. A significant amount of energy has been exerted in addressing yesterday’s threat and fixing today’s vulnerabilities. Today, equal energy is being exerted to identify future threats and prevent tomorrow’s vulnerabilities.
System Security—Shaping or Impeding Systems in the Future?

Ken Kepchar, kenneth.kepchar@incose.org

Systems engineering has made major contributions to many of our transportation systems over the years—from the venerable automobile, to maritime, rail, and aviation systems. Design techniques have been proven over time with some variation depending on the specific mode of transportation. The pace of technology has been evolutionary, even though it varies in application across transportation modes from new and improved materials to taking advantage of miniaturization techniques. However, the evolutionary pace that we have become accustomed to is accelerating rapidly. For example, your next automobile is going digital in a big way, but the design techniques used aren’t necessarily keeping pace with the explosion of digital capability and complexity. Today’s vehicles often contain millions of lines of code controlling functions as diverse as in-vehicle entertainment to safety of life such as antilocking braking. While introduction of digital capability offers a richer functional experience for the driver, at what price does this come? A recent report from Reuters highlighted the risks associated with moving too quickly into a digital auto without fully understanding the implications: “Automakers have so far failed to adequately protect these systems, leaving them vulnerable to hacks by attackers looking to steal cars, eavesdrop on conversations, or even harm passengers by causing vehicles to crash” (Finkle 2012). The article highlighted that, based on a SAE-sponsored study, experts believed the issues affected the entire industry, noting that many automakers use common suppliers and development processes.

In the aviation realm, the United States has embarked on an ambitious modernization of its air-traffic-management system—called NextGen (Next Generation Air Transportation System). We are introducing new technologies, policies, procedures, and systems to increase the capacity, effectiveness, and safety of the National Airspace System in response to changing demands in air transportation. The present system is largely comprised of ground-based, airborne-based, and space-based systems and components that effectively work together to provide the communications, navigation, and surveillance necessary to safely and efficiently transit our airspace. A major portion of the present National Airspace System is based on analog technology; NextGen is moving aviation and Air Traffic Control into the digital age.

An example of this migration is the recent modernization of the navigation systems for the terminal airspace at Newark (New Jersey) International Airport. Approach and departure in and around the immediate airport area was being upgraded to a local GPS-based augmentation system. This was intended to be one of the first production installations of this kind in the US. However, system performance was being intermittently degraded or interrupted during checkout installation testing. This situation was finally traced to mobile GPS interference devices traveling on a major expressway in close proximity to the sensors. This type of signal interference is becoming more prevalent as GPS-based devices proliferate. To address the interference and allow for acceptable system operation in a safety-of-life context, the net result was a system redesign, relocation of the sensors, additional signal protection to the installed system, and a delay in commissioning the system for the better part of a year. This has had a deployment ripple effect of this capability and NextGen across our airspace.

So what’s the point of these two examples? They illustrate that technology drives the decision landscape that systems engineering operates in. As a result, systems engineers need to recognize the shifting importance of digital technologies and adjust their “decision-making portfolios” to accommodate the implications of these new technologies. While these technologies offer impressive benefits to system functionality and capability, they also bring risks that previously did not need to be considered.

In particular, traditional system-development efforts have deferred or ignored security considerations until after the functional architecture has been established and the imbedded concepts are being turned into operational solutions. There seems to be a natural tension between the security community and systems engineering over priorities. This trend has led to a number of commonly held...
security “myths” that need to be dispensed with. Namely:

- Security is a (major) constraint to system performance, not a positive influence.
- Security only drains my budget—I never get resources for it.
- Security is strictly the purview of an expert—not in the systems engineer’s scope.
- Security considerations should wait until the system features and design are settled.
- Security should not influence performance decisions, especially if you want any performance.
- Security technology is out of date before it ever sees the light of day in a delivered product.

Deferring security to late in the development of the system usually forces the resulting security approach to be inefficient at best or even ineffective. It needs to be moved forward in the mainstream design decision process to provide benefit. Consequently, the systems engineers need to include it in the set of technical factors to be balanced during the technical decision-making affecting the final outcome of the system. System security needs to be integrated into the system-architecture-development process.

Architecting is about shaping the future. Because the future involves a great deal of uncertainty, risk is a natural component of any architecting effort. This is amplified in the current fast-moving digital environment where challenges and threats to the integrity, availability, and confidentiality of the system and its information change faster than a system can react. An architecture embodies a structure, a set of relationships, and principles linked to accomplish a purpose. In other words, an architecture establishes a pattern and balance of major elements within some context or environment, shapes behavior through a set of interface relationships, and provides a framework to make decisions.

System security considerations have been relegated to late in the development process because security has traditionally been viewed as an obstacle to system performance. To avoid the common pitfall of treating security as a “necessary evil” that inhibits performance to some degree, system security should be treated as a functional service rather than a constraint. By treating security as an “enterprise” service, systems engineers can now recognize it as a positive contributor to the performance of the system rather than an inhibitor. This allows the security function to be allocated within the system architecture rather than duplicated across the architecture. Systems engineers manage the security “service” in the same manner that they manage reliability, weight, or other “soft” functions.

Effective communication between the systems engineer and security specialist is key to a successful and secure system implementation. The systems engineer should be aware of the limitations of what security solutions offer and their potential limitations on system performance. The system architecture should address distinctions needed for different types of information channels. Information exchanges take place between trusted internal system users and between external semi-trusted users and trusted internal users. Members of the public are restricted, having access only to public websites accessible through the Internet. These interfaces require varying degrees of security mechanisms and careful consideration of the types of information routed across them.

Some basic principles can be employed to ensure effective security integration without the need to substantially change the systems engineering process used for system development and implementation:

Establish a sound security policy as the foundation for design

Security policies are rules supporting sound security practices. Security considerations in the system design need to be driven by the nature of the system and, more importantly, the nature and value of the information the system either generates or has access to. Balancing system protection and performance characteristics is the systems engineer’s job. Protection of the information is usually the end objective, with system protections being provided to allow the system to continue to function at some acceptable level of performance. The information state (e.g., being processed, in transit, or in storage) will frame the extent of security that should be considered. This can be accomplished by prioritizing (categorizing) the information involved. The expertise of the security expert becomes an excellent starting point to accomplish this. However, a “one size fits all” approach to security will probably be counterproductive and meet rejection from the project leaders. The systems engineer should factor in system (and organizational) performance objectives in determining the extent of protection to be considered.

Reduce risk to an acceptable level to limit or contain known vulnerabilities

A basic assumption that should be adopted is that external systems are insecure until proven (and periodically revalidated) otherwise. No security solution is foolproof, and degraded system performance under certain operational settings should be considered, defined, and planned for. The “art” comes in determining where the point of performance unacceptability lies. Security measures that are too strict (more protection) run the risk of unacceptable system performance or usability. Lax (or nonexistent) measures put the system at risk for disruption and potential loss of access or use. Planning for less than ideal conditions is always a prudent way to go.
As a result, the systems engineer should insist that the security specialists pursue a layered approach to security that isolates public-access systems from mission-critical resources. Boundary mechanisms should be used to separate computing systems and network infrastructures and to limit or contain vulnerabilities from damaging or impeding the entire system or connected systems.

Good practices in security solutions go beyond the technical architecture of the system itself.

Part of a solid security policy involves the interaction between the system being designed and the outside world. Practices such as requiring separation of operational duties, or establishment of unique user identities provide effective means to establish accountability for actions that may disrupt operations. However, the systems engineer needs to keep the balance of security with operational ease of use in mind.

Architect security to allow flexibility to adopt new technology, including a secure and logical technology upgrade process.

The threat environment is constantly changing. Protective measures have usually been static in their ability to respond. New techniques, technologies, even policies are constantly appearing. The system should not only be able to detect, but also to react to threats, and prevent damage to the system and resident information. The system architecture should support a design and implementation that provides mechanisms for security forensics and incident reaction (such as an audit trail).

What adjustments should the systems engineer make to effectively include security capabilities as a contributing feature in system design?

Understand the contributions a security expert can make so that security is factored into your technical decision making.

Conversely, understanding the security “lingo” allows the systems engineer to balance what the expert is recommending with other specialties. Their interest is strong security; the systems engineer’s interest is balancing all needs to produce an effective system. The systems engineer also has an obligation to management to portray all the specialties in perspective so that sound technical decisions can be made.

Integrate security into the system’s concept of operations and functional requirements set.

Integrating security early in the process ensures that it will be considered throughout. The systems engineer can assist the security community on the types of contributions that would be most effective throughout the lifecycle. To that end, the author has made effective use of a lifecycle model with security artifacts overlaid on the various phases to illustrate to the systems engineering community what should be provided by the security community and when. The acquisition process especially benefits from this approach, since inclusion of security throughout the supply chain provides a basis to integrate solutions early in the development process.

Security risks are included in project risk register rather than handled separately.

Allocate security functionality across the system architecture in a way that effectively balances any potential trades of security with performance.

Tools and techniques that put complexity and system integration into perspective are essential.

A “roadmap” is an effective means of communication. It portrays the security capabilities envisioned, when they would be available for implementation, how they support planned capabilities and supporting technologies in the system, and when decisions need to be made to keep all aspects of system development on track. Tools of this nature usually take time, effort, resources, and require above all, clarity of goals. As a result the effort should be tailored to fit the needs of the project. Simpler systems benefit from simpler approaches that link the security provisions planned to the performance characteristics of the system involved.

The operational integration challenges involved are constantly in flux.

New systems usually interface within an existing operational environment, touching systems based on different technologies, standards, or security concepts. Hence, legacy integration becomes key, especially if the information handled by the system is safety of life or “critical” in some fashion. If compliance regulations are involved, implementation of security features need to be sensitive to interoperability, a fancy way of saying that systems need to be able to exchange information with each other regardless of the technology or underlying standards involved. This is of paramount importance in the realm of commercial aviation, where airlines operate globally and where each country may have its own set of standards and operational protocols.

As the complexity and technology of systems change, the systems engineer’s perspective needs to adjust accordingly. Shifting system design to include digital-based technologies offers enormous benefits to everyone involved. However, they also introduce different risks than those previously dealt with by systems engineering. As a result, some adjustment in our approach to achieving balance in system design is necessary. One adjustment is to introduce system security at the architectural development stage to provide early insight into the delicate balancing act between the benefits anticipated and the risks incurred in selecting
**What Does a Systems Security Engineer Do and Why Do Systems Engineers Care?**

Janet Carrier Oren, janet.oren@incose.org

Making today’s systems secure is challenging. Systems engineers already have enough to manage without trying to balance this specialty discipline on their own. Systems security engineers can bring the expertise and security unique processes necessary to integrate this discipline with an overall systems engineering effort.

Just as you would not build an airplane without a safety engineer or a bridge without a structural engineer, you should not build an information system destined to be connected to cyberspace without a security engineer. But this continues to occur for critical systems in many commercial and government industries. This article provides the history of the discipline and relates it to safety engineering, a field familiar to systems engineers. Systems engineers are challenged to encourage the growth of this discipline and integrate it into their projects.

**Systems Security Engineering History: 40 Years of Seeking Acceptance**

The concept of systems security engineering was relatively well understood in 1970 to mean independent verification of the acceptability of security controls. However, we find ourselves in 2013 with no generally accepted process for performing this function and integrating it with systems engineering and the system-development lifecycle even though that is what has been envisioned for over 40 years.

The discipline’s path over those 40 years has been fairly consistent, but unlike safety engineering, which took hold in just a few years to resolve the fly-fix-fly model of airplane development, security engineering failures have not had the impact of an airplane crashing. The following is a sampling of the history through several key documents from the United States government. In recent years, commercial texts and articles with similar themes have emerged.

The origin of systems security engineering is generally traced to the 1970 Rand Report, which recognized that security of computer systems needed to be addressed by a team that included security experts. The task force also recognized that “details of an adequate certification procedure . . . are difficult to define” and so no details on this process were provided (Ware 1979). Note the recognition of expertise, having a procedure, and being part of a team.

Twenty years later Jonathan Weiss presented a paper that used the term system security engineering and provided additional comments on the need to integrate security into system development. Why do systems engineers care about systems security? Security is an integral part of the system’s lifecycle and provides overall systems engineering effort.

**Process Model**

![Figure 1. Systems engineering process model (MIL-STD-499B) (National Security Agency Central Security Service 1994)](image-url)

References

development at the National Computer Security Conference. The following year at the conference Dr. Howe recognized that “Information Security is a system problem,” and discussed efforts underway at the National Security Agency to define a process for the US Department of Defense.

The Information Systems Security Engineering Manual, was published for government use only by the National Security Agency in 1994, was aligned with the Department of Defense Systems Engineering Process Model found in Military Standard 499B and shown in figure 1.

Entering the twenty-first century, we finally see public government and commercial descriptions of a systems security engineering process. In 2000 and 2002, the Information Systems Security Engineering Process was included as chapter 3 of the Information Assurance Technical Framework, which was written by a consortium facilitated by the National Security Agency. It also included emphasis on a relationship between systems engineering and systems security engineering. As a final example, the 2008 Engineering for System Assurance Guidebook states that “assurance for security must be integrated into the systems engineering activities to be cost-effective, timely, and consistent” (National Defense Industrial Association System Assurance Committee 2008).

Throughout this history, common themes are maintained. Systems security engineers need to be experts in the field, they need a process to execute, and they need to be integrated with systems engineering.

Where to Go from Here: Experts with a Process

There are many parallels in system safety and system security. We can take advantage of system safety’s 40–50-year head start and learn from their successes in three key areas: expertise, process, and integration. Expertise and process lead to integration.

Expertise of safety engineers is key to the impact of this critical specialty engineering field. If the safety engineer says the airplane cannot fly, then the airplane does not fly. Why do the safety engineers have this power? In a word, expertise.

Developing this level of expertise is why systems security engineering should remain a discipline as opposed to training all systems engineers on security. Expecting systems engineers to be familiar with their roles and responsibilities as well as maintaining knowledge of the constant change in threats and mitigations is unreasonable. Systems security engineers are responsible for studying threat and attack methodologies to develop new mitigations or assess existing ones. The systems engineers should come to rely on the systems security engineers just as they rely on their safety and materials experts.

The field of system safety was developed by groups of experts, led by the United States government, who studied and documented their practices. The commu-

Figure 2. Context diagram for security engineering principles (Howe 1992)
allow for development and application of methodologies to further the field.

The primary government document describing systems security engineering has not been revised since 2002 (National Security Agency Central Security Service 2002). However, a significant update has been in process for two years with the goal of publishing National Institute of Standards and Technology Special Publication 800-160, *Systems Security Engineering*, in 2013. While the process will be more detailed, it is not significantly changed from Dr. Howe’s ideas and the five activities with constant effectiveness assessment shown in figure 3.

The draft revision was based on a study of the systems security engineering process that transformed the 24-page description from 2002 using standard business process modeling techniques, not unlike the systems engineering process models in use today. The result is a more complete definition of a process that includes defined roles, tasks, and outputs. Each task can now be expanded to include methodologies and best practices, thus building upon this new framework for the discipline. Additionally, in support of the desire for integration the “process leverages the concepts, terms, principles, and practices of systems engineering as defined by IEEE 15288 to facilitate consistency in application as part of a systems engineering effort” (National Security Agency Central Security Service 2012).

In my experience with weapon systems, it is clear that managing and executing large efforts requires well-understood processes with common terminology. As a systems security engineer supporting a systems engineering effort, I was able to gain credibility by being able to understand and interact with the systems engineering activities. All descriptions of systems security engineering to date are based on the systems security engineer having basic knowledge of systems engineering as the foundation of their efforts. This facilitates the tie between the disciplines.

**Can Expertise and Process Lead to Integration?**

Specialty disciplines like safety and security do not exist without being integrated into the practice and concepts of systems engineering. After all, what are they trying to secure?

The safety engineer works within cost, schedule, and other program parameters, but is not constrained by them when safety is impacted. System security engineering should not drive how systems are engineered or operated, but it should be integrated to enable business or mission functions.

Like the prior publications, the draft National Institute of Standards and Technology Special Publication 800-160 includes a process and application scenarios to describe how systems security engineering can be applied at any stage in a system’s lifecycle. Also, alignment to Institute of Electrical and Electronics Engineers standard 15288, *Systems and Software Engineering—System Life Cycle Processes*, and the Risk Management Framework used by Publication 800-160 are demonstrated.

Scenarios were written to demonstrate that systems security engineers play a key role in system development as well as system operations and maintenance. When a security incident occurs, the system security engineer is employed to examine systemic causes that could include lack of policies or procedures, physical security violations, and other concerns that are not typically considered in incident recovery.

Nearly every system being developed or used today has a security concern to be addressed. This is easy to recognize in systems in the news such as critical infrastructure systems, including power grids. It might be less easy to recognize in your home computer, but these too are vulnerable and their vulnerability may impact other systems.

Just as products today are safer due to the success of the system safety community, emulating their model can lead to success in the systems security engineering community—*if* the need for system security is accepted and embraced by systems engineers, system operators, and system users.

In the 1950s, “the need for system safety was often motivated by the analysis and recommendations resulting from accident investigations” (Ericson 2006). In 2013, computer attacks are well known and are impacting many systems, including critical infrastructure systems.

The need is clear. The process is imminent. The expert community is growing. The only question that remains is whether the integration will occur. Systems engineers are encouraged to include systems security engineers on their teams to address the security needs of today’s systems.
Addressing Attack Vectors Within the Acquisition Supply Chain and the System-Development Lifecycle

John F. Miller, jfmiller@mitre.org

The security risks for large, complex systems are neither fully understood nor adequately addressed by the systems engineers responsible for system specification, design, implementation, and integration. The threat of malicious insertion and tampering throughout the development and supply of critical components is a broad systems engineering concern for both government and commercial systems. The general nature of the evolving, advanced, and persistent threat is the malicious exploitation of system vulnerabilities in fielded systems. In addition to cyber attacks initiated during system operation, emergent, more complex threat-actor involvement can occur early in and throughout the acquisition lifecycle. By inserting malicious software and counterfeit components during system design and development and across the supply chain, adversaries can gain system control for later remote exploitation or plant “time bombs” that will degrade or alter system performance at a later time, either preset or event-triggered.

This article focuses, from a systems engineering view, on understanding and addressing the “attack vectors” used to exploit vulnerabilities in the system-acquisition supply chain and throughout the system-development lifecycle. Attack vectors can be defined as the routes or methods used by an adversary to exploit systems-design vulnerabilities or process weaknesses to cause undesirable consequences. Examples include the malicious insertion of software into open-source libraries and the substitution of counterfeit hardware components in a receiving department at a lower tier of the supply chain. The former exploits a design vulnerability associated with open-source code and the latter exploits a receiving department process weakness.

Adversarial attacks are composed of many attributes, including the attack origin (or threat source), attack goal (why executed), attack payload (means for exploitation delivered by the attack), attack vector (medium through which an attack payload is delivered or introduced into the system), and attack effect (adverse impact accomplished by the attack). In performing a threat analysis and vulnerability assessment, each potential attack is laid out with a specific path that assesses these factors to inform a risk assessment. This article is focused primarily on understanding the pool of attack vectors within a systems engineering framework, including discussion of payloads and effects.

Figure 1 (Reed 2012) presents a representative set of supply chain attacks. The description of each attack on the right-hand side incorporates both attack vectors and payloads associated with malicious insertion in the supply chain. For example, the second attack in the list combines the attack vector of infiltrating the program office or prime contractor site for malicious insertion with the attack payload of back doors or malicious logic inserted into microelectronics. Points in the supply chain where infiltration can occur are shown on the left-hand side of the figure. Representative attack payloads include modified mission (or business) data, malicious software, and counterfeit components during system design and development.

Representative attacks illustrate where in the supply chain the infiltration occurs and what the malicious insertion accomplishes.

### Representative Supply Chain Attacks

**Program Office**
- Clandestine changes to mission data
- Infiltration of sites to insert back doors and malicious logic into some micro electronics (FPGAs and other devices)
- Infiltration of company receiving department to add/substitute components with backdoors to allow remote penetration during operations, denial of service, etc.
- Infiltration of transportation companies to intercept component shipments (developmental or COTS) and substitute components that have malicious code inserted
- Infiltration allowing malicious software implantation through third party bundling
- Establishment of shell company to insert counterfeit parts
- Infiltration to manipulate the hardware or software baselines
- Infiltration of company software development to insert software which exfiltrates data
- Infiltration to compromise the design/fabrication of hardware

**Contractor**
- Infiltration of site to insert back doors and malicious logic into some micro electronics (FPGAs and other devices)
- Infiltration of company software development to insert software which exfiltrates data
- Infiltration to manipulate the hardware or software baselines
- Infiltration of company software development to insert software which exfiltrates data
- Infiltration to compromise the design/fabrication of hardware

**Distribution Process**
- Infiltration of sites to insert back doors and malicious logic into some micro electronics (FPGAs and other devices)
- Infiltration of transportation companies to intercept component shipments (developmental or COTS) and substitute components that have malicious code inserted
- Infiltration allowing malicious software implantation through third party bundling
- Establishment of shell company to insert counterfeit parts
- Infiltration to manipulate the hardware or software baselines
- Infiltration of company software development to insert software which exfiltrates data
- Infiltration to compromise the design/fabrication of hardware

**Processing/Packaging**
- Infiltration of sites to insert back doors and malicious logic into some micro electronics (FPGAs and other devices)
- Infiltration of transportation companies to intercept component shipments (developmental or COTS) and substitute components that have malicious code inserted
- Infiltration allowing malicious software implantation through third party bundling
- Establishment of shell company to insert counterfeit parts
- Infiltration to manipulate the hardware or software baselines
- Infiltration of company software development to insert software which exfiltrates data
- Infiltration to compromise the design/fabrication of hardware

**Primary Production**
- Clandestine changes to mission data
- Infiltration of sites to insert back doors and malicious logic into some micro electronics (FPGAs and other devices)
- Infiltration of company receiving department to add/substitute components with backdoors to allow remote penetration during operations, denial of service, etc.
- Infiltration of transportation companies to intercept component shipments (developmental or COTS) and substitute components that have malicious code inserted
- Infiltration allowing malicious software implantation through third party bundling
- Establishment of shell company to insert counterfeit parts
- Infiltration to manipulate the hardware or software baselines
- Infiltration of company software development to insert software which exfiltrates data
- Infiltration to compromise the design/fabrication of hardware

![Figure 1. Representative supply chain attacks (figures 1 and 2 generated by Peter Kertzner of the MITRE Corporation)](clifs.php)
counterfeit components, and otherwise manipulated hardware or software baselines. For each of these, the associated attack vectors include many points across the supply chain. For example, malicious insertion can occur, from bottom to top in figure 1, at the primary software developer at the lowest sub-tier of the supply chain, third-party bundlers or adversarial shell companies in the supply chain, during shipment between suppliers and in receiving departments across the supply chain, and during software and system integration at various levels, including infiltration at the prime contractor’s site or even through trusted insiders at the acquirer’s location.

Consider the growing concern of malicious software introduced during its development at a lower-tier primary developer in the supply chain. The identity of such suppliers is often unknown, as are their security practices across the software development lifecycle. Figure 2 illustrates how malicious insertion can occur almost any time during the software-development lifecycle. Representative attack vectors for the insertion of malicious software (including requirements, design, code, and development and test tools) are listed on the right-hand side. The attack vectors describe how the malicious insertion is accomplished at a given point in the lifecycle. The associated phases of the lifecycle are illustrated on the left-hand side.

Understanding attack vectors provides insight into how they should be included and addressed by systems engineers. The next section will examine the intersection of attack vectors with activities of systems engineering using two different frameworks. The first is the standardized set of systems engineering processes defined by IEEE standard 15288 (IEEE 2008). The second framework is a system security engineering methodology developed by the US Department of Defense (DoD) for the protection of mission-critical system components (Baldwin et al. 2012; and the article by Popick and Reed in this issue of INSIGHT). Both frameworks include activities performed by both the acquirer (either a government procurer or a business customer) and the supplier (either a government contractor or a business vendor), depending on contractual and organizational structures and the current phase of the lifecycle. This article does not attempt to distinguish who performs each activity.

Figure 3 lists all the standardized systems engineering processes (IEEE 2008). The processes most relevant to the engineering concerns of attack-vector analysis are shown in italics. The following paragraphs outline the handling of attack vectors within those processes.

**Risk Management**

Risk management is a foundational process for system security engineering. Engineers must consider attack vectors against exploitable system vulnerabilities to identify risks. Attack vectors help frame the risk and inform the risk likelihood determination. Risk assessments consider the attack vectors as the risk’s root cause. Risk mitigation comes into play when a cost-benefit trade-off analysis is used to assess potential countermeasures to mitigate the attacks.
Stakeholder Requirements Definition and Requirements Analysis

From a supply-chain threat perspective, system security engineering can and should be comprehensively integrated into the requirements processes (see Popick and Reed’s article in this issue). Starting early in the lifecycle, engineers performing trade-offs to develop and refine the preliminary system requirements and statements of work should consider security. Specific attack vectors indicate specific countermeasures. These countermeasures should then be specified as security requirements, either as system-design requirements in the system specifications or as process-activity requirements in the statements of work.

Architectural Design and Operation

The system must be designed to defend against unauthorized access, control, and alteration during development and operation. The diversity of associated attack vectors makes it relevant to consider them during the evolution of the system, across all the technical processes. During the maturation of the notional system architecture and design, it is important to consider specific attack vectors targeted at the interfaces between system elements and at system boundaries (including both technical design and supply chain interfaces). As the design evolves, potential countermeasures will be allocated to the most likely and significantly consequential attack vectors. A particular attack vector may indicate a very likely consequence to a critical component, leading to the need to consider alternative concepts and system designs.

Systems engineers will naturally focus on preventing attacks; however, the developing design must also incorporate detection countermeasures (that monitor system performance, capture data about attacks, and alert the user) and response countermeasures (that analyze attacks and mitigate them through resilient system components and altered processes). Throughout the lifecycle, maintaining a forward-looking awareness of specific potential attacks during operations will help ensure that the system design addresses all security concerns while meeting stakeholder performance requirements. Developing the strategy for operation should also account for attack vectors associated with cyber attacks.

Implementation, Integration, Transition, and Maintenance

The attack vectors manifested by insider threats and within the supply chain should be a significant focus of the security-related activities across the full acquisition lifecycle (for both acquirer and supplier). Especially during implementation, integration, transition, and maintenance, the system must be protected from insertion of backdoors, worms, and malicious acts within the supply chain. Associated attack vectors should be considered when identifying constraints imposed on the design solution by the implementation technology and the implementation and integration strategies.

Potential supply-chain diagrams that map the movement of critical components from original equipment manufacturers through all of the intermediary suppliers, subsuppliers, and system integrators (see Popick and Reed’s article for an example) can aid in the analyses of supply-chain attacks during implementation and integration. When recording evidence that implemented system elements adequately address attack vectors, a best practice is to build a security-assurance case (ISO and IEC 2011). When defining the system-transition strategy, adequate mitigation of attack vectors should be included to protect installation within the operational environment and to validate a secure system configuration, as installed. Lifecycle sustainment engineering must include a renewed consideration of design and supply-chain attack vectors to ensure protection during postproduction modifications.

Acquisition

To ensure adequate consideration of attack vectors (and systems security engineering in general), systems engineers should be involved in all aspects of the acquisition process, including establishing the acquisition strategy, preparing solicitations and selecting suppliers, negotiating and monitoring agreements, and determining and verifying acceptance criteria. The acquisition strategy should incorporate intelligence assessments of threats to the supply chain, and the selection of suppliers should include consideration of trusted suppliers for critical components.

Human Resources Management

Human-resources departments across all levels of the supply chain should include processes to counter the attacks illustrated in figures 1 and 2 by identifying the specific security certifications and training needed by systems engineering staff and ensuring qualifications are met prior to assigning them to program staff. Other processes (such as access constraints and remote monitoring of what is accessed) should be established to prevent, detect, and respond to exfiltration attacks.

Configuration Management

Many attack vectors point specifically to countermeasures associated with a robust configuration-management process. Such attack vectors include, for example, malicious software implantation of “time bombs,” hardware or software baseline manipulations, and hidden back doors to gain unauthorized remote access. A configuration-control system that includes least user access, robust authentication, and two-person inspection and approval of changes can help counter such attacks.

Information Management

Attack vectors targeted at the exfiltration of critical information about systems,
their development, and associated technology, as well as critical processes and capabilities, are addressed by information-assurance practices. Most of the activities of the information management process are accomplished through information-assurance controls to ensure integrity, availability, and confidentiality. In addition, it is important to incorporate protections that nullify attack vectors aimed at compromising information and data that is stored, processed, and transmitted by the system during operations.

The US Defense Department’s Methodology

Although system security engineering traditionally has been viewed as a specialty engineering area, the 15288-based discussion conveys that implementing it to address adversarial threats must be tightly integrated within a holistic systems engineering approach. The US defense department has engaged in a number of efforts to assure trusted systems and networks against an advanced persistent threat that continues to flourish. In particular, they have developed a system security engineering methodology that provides a defined set of activities and analyses carried out by a multidisciplinary team led by systems engineers to identify and protect mission-critical system components (Baldwin et al. 2012; Popick and Reed 2013). The methodology is a general one with broad applicability to all engineered systems, both government and commercial. It is built upon standard systems engineering processes (e.g., risk management and requirements definition) as well as traditional security practices (e.g., threat analysis and vulnerability assessment).

A threat analysis within the system security engineering methodology identifies potential threat events that can be executed at any time during the lifecycle, from concept refinement and requirements definition through maintenance and disposal of the system. The general nature of the threat is known, but considering specific attack vectors helps illuminate when, where, how, and by whom an exploitation might occur. Key input is provided by both supply chain threat intelligence assessments and system-specific analyses. Threat analysis is an iterative activity which is repeated across the lifecycle as the system evolves, as new information is available, and as the engineering-based risk–cost–benefit tradeoffs are refined.

The Department of Defense sponsored a generic threat analysis in which it collected threat information from several sources and applied associated attack vectors across the lifecycle and supply chain. Figures 1 and 2 exemplify the results of that analysis (Reed 2012). A refined program-specific and system-focused threat analysis then identifies and analyzes the potential significance of attacks targeting each critical component, according to the unique attributes of that component, as well as system-specific (security) requirements, program-dependent and operation constraints, and intelligence assessments of the threat.

A vulnerability assessment within the defense department’s methodology considers attack vectors and access paths to determine opportunities for adversarial exploit of both process and design weaknesses. Attack vectors are the means by which an adversary exploits vulnerabilities. Initial vulnerability assessments consider the attack vectors and access paths at a high level (figures 1 and 2). A refined system-focused vulnerability assessment incorporates the system architecture, concept of operations, supply-chain diagrams that map alternative critical component acquisition paths, and contextual information associated with specific attack vectors and payloads to identify system-specific (component and interface) vulnerabilities. Potential architectures and supply-chain diagrams may indicate the need for trusted suppliers of certain critical components or the need for a redesign effort.

Attack vector analysis is key to robust system security engineering. Ongoing efforts by engineers and security professionals within several subdisciplines of system security address threats, vulnerabilities, and attacks at various levels. For example, the US National Institute of Standards and Technology recently updated and enhanced its guide for conducting information security risk assessments (NIST 2012). The guide describes threat events targeted at information systems and provides a compilation of representative examples of adversarial threat events.

In another example, the US Department of Homeland Security is sponsoring an ongoing effort to grow and maintain a publicly available catalog that provides a collection of common attack patterns (the “common attack pattern enumeration and classification,” CAPEC) of typical methods for exploiting software (MITRE Corporation 2012). The attack patterns of CAPEC capture and communicate the software attacker’s perspective. The attack patterns are derived from the concept of design patterns applied in a destructive rather than constructive manner and are generated from in-depth analysis of real-world software exploits.

Similarly, the MITRE Corporation has developed a Threat Assessment and Remediation Analysis (TARA) methodology to identify and assess cyber threats and to select effective countermeasures (Wynn et al. 2011). The TARA methodology relies on a catalog of adversarial tactics, techniques, and procedures mapped to countermeasures for threat remediation and risk reduction. The current catalog has been built primarily from engagements with information systems programs.

Building on these and other known sources for system security engineering, the US Department of Defense sponsored a systems engineering-led effort to develop a catalog of attack vectors and countermeasures. Figures 1 and 2 were generated from a high-level synthesis of the evolving DoD catalog to demonstrate the vectors’ breadth as well as the supply chain (figure 1) and software lifecycle (figure 2) contexts within which the vectors are executed (Reed 2012). Continued enhancements of the methodology will lead to a refinement of how attacks are realized and miti-
Requirements Challenges in Addressing Malicious Supply Chain Threats

Paul R. Popick, paul.popick@incose.org; and Melinda Reed, melinda.reed@incose.org

In today’s environment of cyber attacks and exploitation of system vulnerabilities, the systems engineer needs to be more aware of security during the system specification and design stage. Recent examples of supply chain attacks include computer motherboards shipping with malware, military chips from China with secret backdoors, and a bank employee inserting malware into the ATM network.

This article discusses the US Department of Defense (DoD) state of practice for incorporating trusted system and network security requirements into the specifications for large, complex systems. The article describes the current environment, the trends that are influencing the need for system security engineering, and the types of system security requirements and analysis techniques the DoD is using. This article updates the system security engineering risk-cost-benefit trade-off analysis described in previous papers (including Baldwin et al. 2012).

The trends that are contributing to the system-security challenges facing major DoD programs include the increasing reliance on commercially available technology, complex supply chains that include thousands of suppliers worldwide (figure 1), system interconnectedness, and the identification and exploitation of the supply chain and commercial off-the-shelf (COTS) vulnerabilities.

The complexity of supply chains and development processes of major acquisition programs (with prime contractors, subcontractors, suppliers, and subsuppliers) makes it difficult for anyone truly to know what is in the system and where it came from. Many of the COTS products have complex supply chains that are not secured to prevent alteration and malicious insertion. In addition, open-source code and code of unknown origin are often incorporated into the system’s COTS components and the COTS tools used to develop DoD subsystems. These COTS and open-source products are widely available for study, reverse engineering, and exploitation of vulnerabilities.

The systems engineer and system security engineer must consider not only the security of the system but also the security of the supply chain (see John Miller’s article in this issue), the COTS products used in the system, and the information incorporated into the system as much of the development and manufacturing exist outside of traditional controls. In designing and trading off potential components, the systems engineer must consider whether the COTS products are vulnerable to attack within the supply chain, the development environment, the development process, the system maintenance process, and the operational system.

The motivations for exploiting these vulnerabilities include financial gain, exfiltration of data, denial of service, and alteration of mission results. For a further discussion of how attack vectors are linked with the vulnerability assessment and how attack vectors inform the requirements analysis, see John Miller’s article.

Figure 1. Global complexity of DoD supply chain
Stakeholder needs are captured in the system-requirements documents from the sponsor and from applicable DoD directives and instructions (Kendall 2011; DoD 2012). The related DoD directives and instructions require that systems incorporate program protection, information assurance, protections related to the supply chain, counterfeits protections, and anti-tamper. These policies do not describe the details of the protections required, allowing the systems engineer and the systems security engineer the flexibility to define the specific requirements and design.

Security Analysis Trade-Off Method

The systems engineer and system security engineer analyze risk to determine appropriate trade-offs between security protection requirements and technical performance, cost, and schedule requirements. The systems engineer needs to recognize that vulnerabilities will continue to be identified during the system development and operation, and thus the system security requirements will need to be reassessed and updated as system requirements and design decisions are made. Regardless of the robust protection functions a program may incorporate to prevent attacks, the systems engineer and system security engineer also need to consider how to respond to an attack that penetrates the system. The systems engineer and system security engineer will need to incorporate functions that not only prevent but also detect and respond to attacks that exploit vulnerabilities.

To aid the systems engineer and system security engineer to analyze system security and to make trade-off decisions, the DoD has begun using an updated risk–cost–benefit trade-off analysis method for trusted systems and network security shown in figure 2. Note that the risk assessment depicted uses the criticality analysis for the consequence factor and a combination of the threat and vulnerability assessment as the likelihood factor.

Program managers and systems engineers apply this system security analysis method before each systems engineering technical review and periodically during the operations and maintenance phase of the DoD acquisition lifecycle. These updates of the system security analysis ensure that the program includes security updates to the system requirements and design characteristics that align with other updates as a result of elaborating the system. The method also promotes consistent system security engineering analysis across DoD programs as well as within a program. Figure 3 shows the points for systems engineering technical review in the DoD lifecycle where the system security analysis updates are incorporated into the requirements and design baselines.

![Figure 2. Risk-cost-benefit trade-off analysis method](image)

**Table 1. Protection failure criticality levels**

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level I</td>
<td>Total Mission Failure</td>
</tr>
<tr>
<td>Level II</td>
<td>Significant Mission Failure</td>
</tr>
<tr>
<td>Level III</td>
<td>Partial/Acetable Mission Failure</td>
</tr>
<tr>
<td>Level IV</td>
<td>Negligible Mission Degradation</td>
</tr>
</tbody>
</table>
Threat and Vulnerability Assessments

DoD systems are exposed to threats of malicious insertion and tampering throughout the development and supply of critical components from external and internal sources. This exposure is further exacerbated by the use of a significant number of COTS parts that are obtained through a global supply chain. Examples of malicious insertion threats are widely publicized and include telecommunication switches that exfiltrate data and radar systems that are unable to detect a particular country’s planes.

The vulnerability assessment identifies weaknesses in system design, development, production, components, operation, and the supply chain that can be exploited to prevent or degrade the system’s operation. During the requirements analysis, systems engineers evaluate potential vulnerabilities to critical function components to determine whether additional security requirements or constraints are needed to mitigate vulnerabilities. Identifying vulnerabilities extends the typical engineering process beyond the system to also consider the protection of the supply chain and the development environment. Systems engineers analyze the potential for the components to be exploited or subverted during development and supply, and they consider the potential to design in resiliency to allow the system to detect exploitation and continue to operate.

Early in the system-acquisition process, systems engineers need to identify potential vulnerabilities by examining the system concepts and critical functions for access paths. One approach is to list common vulnerabilities of the system, supply chain, and development environment, drawn from industry databases (SEI 2012; Mitre Corporation 2012) and the Defense Acquisition Guidebook (DAU 2012, chapters 4 and 13). Engineers can use this list to evaluate whether the requirements preclude these vulnerabilities. Another approach is to draw upon information-assurance and systems-security-engineering expertise to identify possible attack vectors and then use the attack vectors to determine whether the requirements prevent the attack. A vulnerability is listed for those attacks that are not prevented by the current set of requirements.

An analysis tool that DoD has used with both of these approaches is to draw a map of the movement of a critical component from the original equipment manufacturing through all of the intermediary contractors to the prime contractor showing the company name and the site location (figure 4). This map helps the program identify vulnerabilities with each link in the supply chain. The vulnerability analysis results are used as part of the risk assessment to determine the likelihood of losing mission capability (figure 2).

The information-assurance assessment is a specialized vulnerability assessment that uses the system categorization along with the required baseline controls to identify confidentiality, integrity, and availability vulnerabilities to the system and the critical functions that are not prevented by the baseline controls. The results of the information-assurance assessment are combined with those from the vulnerability assessment to inform the risk assessment and assist with determining the strength of implementation required, tailoring of the control set, and translating the controls into requirements. The systems engineer and system security engineer need to ensure that threat, vulnerability, and information assurance assessments examine the findings from one another to avoid missing or duplicating vulnerabilities. Any previously identified vulnerabilities are used as part of each of the assessments. The systems engineer and system security engineer examine the system concept and requirements to determine the set of potential baseline and additional information assurance controls necessary to mitigate these risks to an acceptable level. The information assurance controls are defined by requirements and specific design details necessary to ensure they mitigate the identified confidentiality, integrity, and availability vulnerabilities. These mitigation requirements are captured in the system requirements, functional baselines and process requirements in the Statement of Work.
The results of the criticality analysis, vulnerability assessment, threat assessment, and information-assurance assessment contribute to the risk assessment. Countermeasures are cost-effective activities and attributes to mitigate or neutralize threats to and vulnerabilities of the system-critical functions and associated components. They vary from process requirements to system requirements, constraints, and design attributes. Although potential countermeasures are often identified as part of each of the assessments, during this step the systems engineer develops a comprehensive list of potential countermeasures. The potential countermeasures list needs to include countermeasures that detect and respond to attacks as well as prevent the attacks.

For example, a system-detection countermeasure may be a function that is built into the system that identifies when a critical function is behaving in an unauthorized manner. It sends an alert and logs relevant data to allow for later forensic analysis. Similarly, a process-detection countermeasure may be one that limits update or insertion of software code, sends alerts about unauthorized access attempts, and logs data for later forensic analysis. A “respond” countermeasure determines how the system or the supply chain process reacts to an attack. The “detect” and “respond” countermeasures ensure that awareness and response capability are built into the system and its supporting processes.

Risk-Cost–Benefit Trade-Off

The risk–cost–benefit trade-off analysis includes two levels of trade-off analysis. The program conducts an analysis within the security domain to trade off the potential countermeasures to identify a cost-effective set of system security requirements. The other trade-off level considers the broader system functional and nonfunctional performance requirements and design characteristics to ensure a balanced trade-off of system security requirements versus performance and cost impacts. For example, a security countermeasure to monitor a critical function’s behavior may lead to an unacceptable decrease in the function’s throughput or response time. Similarly affordability of the system requirements may also necessitate examination of alternatives requirements. This leads to a dynamic environment in which systems engineering trade-off results outside the security domain trigger a need to update the system security engineering analysis and trade-offs.

Risk, cost, and benefit factors influence these two levels of trade-offs. The systems engineer may explore alternative designs to evaluate the new or revised requirements. The output of this step is a set of affordable countermeasure requirements to be incorporated into the system requirements baseline and acquisition-process requirements from the Statement of Work.

Future Plans

The defense department is just beginning to use this trusted systems and network analysis method for system security engineering. The method provides an objective way of analyzing and quantifying the system security and developing the system security requirements. Extending the system security engineering trade-off analysis into the supply chain, development processes, and the development tools requires systems engineering interactions with procurement and acquisition processes that are not normally employed during the system specification and design.

The need to address global supply-chain threats and development threats has made it necessary to implement in parallel with the development of system security engineering methods and tools. This concurrency leads to some confusion by the system security engineers as the methods and tools are continuously upgraded. DoD is developing an outreach and training program to ensure that the systems engineers and system security engineers are trained to perform this work.

Programs are finding it challenging to respond to changing supply-chain threats, development threats, and uncovered vulnerabilities. Using this method before each of the systems engineering technical reviews and periodically during operations may assist programs to respond to this challenge. This method emphasizes the affordability considerations through the cost–benefit trade-off to ensure that system security requirements are part of the overall acquisition and fielding of secure operational systems. In order to fully address supply-chain issues, the systems engineering community needs a comprehensive outreach approach to increase leadership awareness and to train program managers, systems engineers, and system security engineers. Professional societies, industry associations, and industrial firms have an important role to play in this outreach.

The Department of Defense has developed guidance for the Defense Acquisition Guidebook chapters 4 and 13 (DAU 2012) and has prepared awareness briefings for the acquisition community. This guidance has increased awareness of the need for system security engineering training for systems engineers and system security engineers. The DoD is developing training material and will incorporate it into courses at the Defense Acquisition University as well as continuing education courses offered through industry and professional organizations.

Information-assurance controls are documented and have been in use within the DoD for a number of years (US DoD 2003). The information-assurance control policy is currently being updated and will be issued in the near future to include supply-chain controls and a risk-management framework for cybersecurity (DoD, forthcoming). In systems engineering terms, the information-assurance controls need to be refined into system requirements because these controls are not described in sufficient detail to evaluate their effectiveness with respect to specific
attack vectors or to specify a system for acquisition. Unfortunately in the past the information-assurance controls have not always been refined and incorporated into the system requirements. This can result in missing or overlapping requirements. The DoD is emphasizing the role of the system security engineering to ensure that the information-assurance controls are refined and incorporated into the system and process requirements.

The Systems Engineering Research Center and other federally funded research-and-development centers have initiated research into secure design methods for the operational system (SERC 2012; SEI 2009). Research is also needed to define secure design methods and process descriptions for the supply chain similar to those for the operational system. To date, the acquisition community has engaged in limited activity (DoD 2010) that has broadly defined the secure supply-chain approaches but has not defined them to the level of detail necessary distinguish between implementations. The DoD is sponsoring activities to begin developing catalogs of these supply-chain methods and is encouraging more industry research into secure supply-chain and software-assurance techniques.

References


———. 2012. DoD Instruction 5200.44. Protection of Mission Critical Functions to Achieve Trusted Systems and Networks.


Ruth Duggan, rduggan@sandia.gov; and Mark Snell, mksnell@sandia.gov

Every system has a reason for existing. The system mission can be stated as a job or function it must perform and is often phrased using system requirements. At its most basic level, systems engineering is the defining of requirements, designing and implementing systems to meet those requirements, and evaluating how well these requirements are met over the system’s lifecycle.

A key responsibility for the systems engineer is to identify and mitigate all significant risks to the system-of-interest during each phase of its lifecycle. Mitigating cost risks and safety technical risks can be difficult enough to manage even with ample amounts of historical data; by comparison, how can a systems engineer effectively manage security risks to the system when security risks occur due to the potential for malevolent, adversarial acts that are difficult to predict?

There is an increasing need for the systems engineer to characterize and mitigate security risks. Whether the system is a business that faces crime and perhaps terrorist attack, or a computer network that needs to operate in spite of cyber attacks, successful adversarial acts may cause the system to be unavailable for long periods of time and result in collateral effects such as business or market losses and even injury or death to users or the public. By better understanding these security risks and their significance during the requirements-development phase, the system engineer and other stakeholders can make more informed choices about controlling such risks over the lifecycle.

Two important technical processes are available to the systems engineer for defining security requirements. First is the stakeholder-requirements-definition process to develop security requirements in the first place. Second is the requirements-analysis process that balances and prioritizes security and other requirements and then transforms them into a functional and technical system description that meets these requirements.

Several key activities can proceed during the stakeholder-requirements-definition process. First, the systems engineer can engage one or more security systems engineers as security stakeholders to address the types of security concerns associated with the system. This system security engineer can then help the systems engineer determine whether or not security is a critical system requirement, based upon unacceptable consequences of system failure.

There is also a reciprocal need for education, first of the systems engineer about security and simultaneously of the system security engineer about the system under design. Some basic topics for educating the systems engineer are discussed next.

Security Principles

When we system security engineers work with systems engineers and other stakeholders, we typically teach them how to think about security and associated risks. Security risks are typically represented as a combination of likelihood of a successful malevolent act by an adversary and the consequence of that act. Thus, it is important for systems engineers to understand what is meant by such terms as threats, adversaries, and consequences of adversarial actions that are relevant to their system-of-interest.

Systems exist within an environment exposed to normal, abnormal, and malevolent conditions. Security events are considered malevolent in that humans cause them with deliberate intent; examples include theft of system assets, sabotage of the system, hostage situations, protests, and denial of service. The threat refers to these types of events. Associated with these events are threat actors, or adversaries, entities with the intention and capability to perpetrate the malevolent acts. Adversaries may be partitioned into insiders who have ability to bypass or control over parts of the security system, and outsiders who do not have such capabilities but may have technical expertise or numbers of attackers that the insiders do not possess. The likelihood of a successful malevolent act is then the likelihood that a threat actor attempts that type of malevolent act over some period of time multiplied by the probability that the actor succeeds, in spite of security, in carrying out that act. The security requirements then define how the system should respond to these threats separately and in collusion.

Typically, security-consequence analyses are performed for similar reasons as safety analyses. Such analyses help clarify the importance of security to the mission and how other areas of the system, such as safety systems, can be manipulated by malevolent human actions to bring about mission failure. Security may be essential to mission success and an integral part of system performance, or it may be a feature of the system added after all other requirements are considered. Consider the difference between the security of a computing system protecting sensitive information and security of a classroom protecting pencils from theft. Consideration of security design principles can also benefit both the system security engineer and the systems engineer when setting security requirements (see Garcia 2001, 58–62).
Complications in Defining Stakeholder Requirements

While this process is conceptually easy, performing this process for security may be complicated. Efforts by the system security engineer and the systems engineer can benefit when these complications are kept in mind.

For example, stakeholders may be either difficult to reach, in which case few requirements are defined, or have constraining requirements to be met. Hard-to-reach stakeholders may be excluded from discussions about the system’s security or may not know enough to realize that they should seek participation. Not properly accounted for, these stakeholders may, in the end, have a more negative impact on the system than the security attack (consider, as an analogy, the effect of the safety event at Fukushima on nuclear power worldwide).

Alternatively, a regulator may have specific requirements that must be met before the system can be produced and deployed. Specific requirements, while clearly defined, may constrict consideration of options during requirements analysis, especially when difficult to balance with other requirements. Also, regulatory requirements may not keep up with advances in technology. For example, prior to 2011, international recommendations for nuclear security systems did not mention computer security (see International Atomic Energy Agency 1994). In 2011, a newer version of the same document was released that did cover computer security (see International Atomic Energy Agency 2011, 21 and 32).

Some key stakeholders may not clearly understand how to interpret security risks and requirements. Just after the September 11 attacks in 2001, many organizations were worried about similar mass-casualty attacks by Al Qaeda. It took some effort to convince them that they were likely not high on Al Qaeda’s list of targets and that they could adequately control their risk against such attacks by having contingency plans in place for such events. At the other end of the spectrum were high military officials who came to value having marked parking spaces and corner offices as emblems of their success; in the mid-1990s it was not obvious to them that these observables could increase their risks of attacks by terrorists or disgruntled personnel.

The systems engineer should also be aware that the complex nature of security itself could cause difficulties in developing security requirements. There are a number of complicating factors that can make it difficult for the systems engineer to integrate security concerns into a holistic view of security risk to the system, as the following paragraphs explain.

Security-system design must address a broad range of security domains—physical, operations, information and cyber, personnel, communications, and survivability—each compartmentalized in the sense that each protects against different threats. “Threats to be considered include conventional, electronic, nuclear, biological, chemical, and other weapons, as well as terrorism or sabotage” (Haskins 2011, 315).

The variety of systems to be protected requires different countermeasures. Smaller, inanimate targets will require different security measures than human beings (such as a head of state) or virtual targets in information systems. The size of the target or its operational environment also influences security system design. Protection systems for a building or vehicle are different from protection systems for a vast infrastructure such as the electric power grid or a business’s enterprise computing network.

Security itself has many elements that may need to be addressed such as people, information, procedures, equipment (physical, spatial, virtual), access control, and need-to-know-based privileges required for access to sensitive information. This complexity often requires that experts be hired to cover each of these areas as well as systems engineers who are able to develop requirements across these fields.

The evolution of threats and incorporation of multiple generations of security technologies over the lifecycle of a long-life system, stretching over decades, poses a particular challenge when the requirements call for upgrades to legacy systems. The systems engineer needs to consider potential advances in the mission system as well as the security system. Consider designing an airport in 1965: what security threats did that designer consider at the time? Even if the systems engineer did somehow foresee future changes in the threat, how could they anticipate the phenomenal technological changes since then?

These complications can make it difficult for the systems engineer to integrate security concerns into a holistic view of security risk to the system and, based on this, determine whether security is a critical system requirement or not. As a result, security can also be at a disadvantage when systems engineers determine what system performance parameters are critical for system success. Unfortunately, the increase in many threats, over time, has gradually reduced this latter concern somewhat for some threats, such as high-consequence terrorist attacks against prominent buildings. An expert system security engineer can help the systems engineer navigate these issues and, based on experience, can suggest methods for defining security requirements so that the resulting system will have a measure of robustness against future threats and technical advances.

Performing Requirements Analysis

Once requirements have been defined, the designer performs a requirements analysis with the stakeholders to resolve conflicts between requirements and transform them into a functional and technical system description meeting these requirements. The systems engineer benefits if he or she understands how security
is ranked within the mission relative to the other types of requirements.

Because of the regard for human life, security is often prioritized lower than safety requirements (consider the conflict between locking security doors and emergency egress). Additionally, security risks can be imperfectly understood by decision makers who then put convenience ahead of security (as in a facility where the reactor operators wanted a bridge over the perimeter intrusion system so that they could get directly to the control room).

Opportunities for jointly mitigating common consequences should be sought rather than partitioning the risks of product or system failure, natural or accidental safety risks, and security risks. Examples of this include selecting materials that are less attractive to adversaries or employing laws of nature and other “first principles” to robustly mitigate a significant consequence across many threats. Additionally, redundant capabilities for continuing a system’s mission, given failure of a protection system, makes the system more resilient.

While some security needs can be handled through technical controls, security is highly dependent on the people associated with the system. Administrative controls such as policy and procedures also play an important role in security. These controls need to be considered by the security system engineer through all the phases of design, implementation, and operations so the role of people in the security of a system is understood and a security culture is fostered.

Robust System Security in the Face of Uncertainty

Systems engineers have a responsibility to address future security risks in developing and implementing systems. Experienced system security engineers know how to consider historical data and predict threat scenarios based on unacceptable consequences. Involving them as early as possible within the system lifecycle can properly engage stakeholders in the identification of unacceptable consequences that will be used as the basis for cost–risk tradeoff analysis.

References


How can we field systems with long life expectancies, embedding security that can deal with attacks that won’t be invented until a year or more after systems are put into service? This article offers a path based on an agile architecture. But first, a quick review of a recent incident will set some context.

The now infamous Stuxnet attack (Kushner 2013) ushered in a new era of attack sophistication, with four vulnerabilities exploited that had not been used before, so-called zero-day attacks. The victim system was Iran’s Natanz nuclear-fuel enrichment plant, with thousands of COTS (commercial off-the-shelf) centrifuges and many SCADA (supervisor control and data acquisition) COTS computers that controlled them. The complete story may never be known, but a more than sufficient portion of the attack method has been reverse engineered, and is now in the public domain—for use by others less resourceful.

There was an “air gap” between the Natanz plant systems and the outside world—it was an isolated internal network with no external network connections. There was apparent confidence that the air gap would prevent an intrusion. But insider folly or insider maliciousness appears to have introduced an infected USB device from the outside. Then, too, there were SCADA computers purchased off the shelf, and placed in a network of communicating devices that likely trusted each other. Even if the SCADA equipment was deeply examined upon arrival, an infection occurring after installation could spread among naïvely trusting networked equipment, unquestioned.

Note that Stuxnet successfully attacked a cyber-physical system, targeting the centrifuges, not the information system. Once implanted, Stuxnet laid dormant most of the time, coming to life once a month for a short period of time, instructing the centrifuges to spin at rates outside their specifications, damaging them over a period of time. While active, Stuxnet spoofed the monitoring system just like the television spy shows, showing normal operating data to the monitors that didn’t reflect what was actually occurring. This slow, stealthy, periodic attack over many months just looked like accelerated wear, and didn’t cause an immediate investigation alert.

How can we mitigate the inevitable never-seen-before threats to fielded systems? Here’s how.

System Architecture is Where Security Enablement Starts... or Stops

Architecture focuses on the high-level allocation of responsibilities between different components of the system, and defines the interactions and connectivity between those components. Responsibility for security requirements established during the
requirements processes should be allocated to functional components and security-specialty components as appropriate during the architectural design process.

System resilience permits a system to operate while under attack, and to recover afterwards. The system may end up with degraded performance, but it will continue to deliver its critical functionality. Resilience is an architectural feature that is difficult to provide later in the development lifecycle, and very costly after deployment.

Long-life systems will have functional upgrades and component replacements throughout their life. Insider threats and supply-chain threats may manifest as components with embedded malicious capabilities that may lie dormant until activated on demand. This argues for self-protective system components that distrust communications and behaviors of interconnected components, rather than relying on system-perimeter protection or trusted environment expectations.

The Concept of Operations

Continuous evolution of system security is necessary to maintain parity with a continuously evolving threat environment. This requires the ability to respond effectively under unpredictable and uncertain circumstances, as often as necessary.

Enabling continuous evolution of system security requires an agile-system concept of operations, one that recognizes the need for effective asynchronous changes to system security. A fundamental agile architecture pattern enables this, and will be recognized in a simple sense as a modular drag-and-drop, plug-and-play architecture, with some critical aspects not generally called to mind with the general thoughts of a modular architecture.

**Need** — A cross-industry study for the US Office of Naval Research in 1991 (Nagel 1992) observed that technology and the environment in which it is deployed were co-evolving at an increasing rate, outpacing the adaptation capabilities of most organized human endeavors. Agility, as a systemic characteristic, was identified as a new need generally missing in systems we call the enterprise, the systems supporting the enterprise, and the systems produced by the enterprise. The decreasing relevance and life expectancy for traditional systems of all kinds create the countervailing need for agility.

**Definition** — Agility is the ability of a system to thrive in an unpredictably evolving environment; deploying effective response to both opportunity and threat, within mission.

**Metrics** — Effective response has four metrics: timely (fast enough to deliver value), affordable (at a cost that can be repeated as often as necessary), predictable (can be counted on to meet the need), and comprehensive (anything and everything within the mission boundary).

**Value Proposition** — Risk management in an evolving unpredictable environment is the value proposition for agile systems. An agile system is constructed to enable and facilitate augmentation, reconfiguration and scalability of reusable assets in response to unpredictable situations. Agility is sustained with active management of responsibilities that constantly evolve the agility-enabling capabilities.

**ConOps** — In short, the system ConOps should call out the ability to reconfigure and augment system security throughout the development and operational lifecycle of the system, and it should call out the need for rapid reconfiguration of security at the system level (or both the system-of-systems level and the system level in a system of systems). Figure 1 depicts a notional architectural concept.

---

### Figure 1. Notional concept—common security infrastructure enables rapid evolution; other security modules are likely to include (for example) functional-module behavior monitoring

---

### Fundamentals

This section focuses on fundamental needs, definitions, and necessary and sufficient enabling concepts for agile systems of any kind (Dove 2001, 2012)—most especially agile security functionality.

- **Architecture** — Achieving good agile response metrics is enabled or hindered by architecture. One fundamental agile architecture pattern has emerged from extensive investigation and appears both necessary and sufficient. It will be recognized in a simple sense as a loosely coupled, drag-and-drop, plug-and-play architecture, with some critical aspects not generally called to mind with the general thoughts of a modular architecture. There are three critical elements in the architecture: a catalog of drag-and-drop encapsulated modules and the module pools in which they belong, a catalog of the passive infrastructure rules and standards that enable and constrain plug-and-play operation, and the designation of an active infrastructure of four specific responsibilities that sustain agile operation.

- **Modules** — Modules are self-contained encapsulated units complete with
well-defined interfaces that conform to the plug-and-play passive infrastructure. They can be “dragged and dropped” into a configuration of response capability, with relationship to other modules determined by the passive infrastructure. Modules are encapsulated so that their methods of functionality are not dependent on the functional methods of other modules, except as the passive infrastructure dictates.

• Passive Infrastructure — The passive infrastructure provides drag-and-drop connectivity between modules. Its value is in isolating the encapsulated modules so that unexpected side effects are minimized and new operational functionality is rapid. Selecting passive infrastructure elements is a critical balance between requisite variety and parsimony—just enough in standards and rules to facilitate module connectivity, but not so much to overly constrain innovative system configurations. Passive infrastructure typically evolves, although slowly and most often when migration to next generation capability is appropriate.

• Active Infrastructure — An agile system cannot be designed and deployed at a fixed event and then left alone. Agility is most active as new system configurations are assembled in response to new requirements—something which may happen very frequently, even daily in some cases. Four responsibilities are required and must be designated and embedded within the system to ensure that effective response capability is possible at unpredictable times.
  - Module Mix/Evolution — Who (or what process) is responsible for ensuring that new modules are added to the roster, and that existing modules are upgraded, in time to satisfy response needs?
  - Module Readiness — Who (or what process) is responsible for ensuring that sufficient modules are ready for deployment at unpredictable times?
  - System Assembly — Who (or what process) assembles new configurations when new situations require something different in capability?
  - Infrastructure Evolution — Who (or what process) is responsible for evolving the passive and active infrastructures as new rules and standards are anticipated and become appropriate?

Figure 2 depicts a graphical representation of the architecture.

**Principles** — Ten reusable-reconfigurable-scalable design principles add the flesh to the bones of the architecture, and are briefly itemized here.

**Reusable Principles:**
- Encapsulated Modules — Modules share well defined interaction and interface standards, and are easily inserted or removed in system configurations.

**Facilitated Interfacing (Plug Compatibility)**—Modules are reusable and replicable, with supporting facilitation for finding and employing appropriate modules.

**Facilitated Reuse**—Modules are reusable and replicable, with supporting facilitation for finding and employing appropriate modules.

**Reconfigurable Principles:**
- Peer-Peer Interaction — Modules communicate directly on a peer-to-peer relationship, and parallel relationships are favored, rather than sequential relationships.
- Distributed Control and Information — Modules are directed by objective rather than method, decisions are made at point of maximum knowledge, and information is associated locally but accessible globally.
- Deferred Commitment — Requirements can change rapidly and continue to evolve. Work activity, response assembly, and response deployment is deferred to the last responsible moment, to avoid costly wasted effort that may also preclude a subsequent effective response.
- Self-Organization — Module relationships are self-determined where possible, and module interaction is self-adjusting or self-negotiated.

**Scalable Principles:**
- Evolving Standards — Passive infrastructure standardizes intermodule communication and interaction, defines module compatibility, and is evolved by designated responsibility for maintaining current and emerging relevance.
• Redundancy and Diversity — Duplicate or replicable modules provide capacity right-sizing options and fail-soft tolerance, and diversity among similar modules employing different methods is exploitable.
• Elastic Capacity — Modules may be combined in responsive configurations to increase or decreased functional capacity within the current architecture.

Requirements — In addition to the system functional requirements, response situation analysis helps identify response requirements that inform the design of architecture, indicating the necessary nature of modules and module pools, which in turn help identify the necessary nature of both passive and active infrastructure. Requirements arising from response situational analysis may not be directly present in customer requirements, but are necessary for effective architecture design. Unlike functional requirements typically captured in all-encompassing specific “shall” statements, response requirements need only enumerate sufficient diversity to result in a capability that can respond to un-enumerated situations. An effective framework for structuring response situational analysis drives analytical thinking in four reactive and four proactive domains. For brevity the framework below provides abstractions without examples. Detailed examples can be found in Dove (2001, 2012), with general coverage of case-making in Sillitto (2013). Note that response requirements are system operational-time requirements, not system design-time requirements; and should be stated as operational needs independent of possible solution strategies which will evolve with time.

Proactive Domains:
• Creation/Elimination—What range of opportunistic situations will need modules assembled into responsive system configurations; what elements must the system create during operation that can be facilitated by modules and module pools; what situational evolution will cause obsolescence of modules which should be removed?
• Improvement—What improvements in system response performance will be expected over the system operational life?
• Migration—What evolving technologies and opportunities might require future changes to the infrastructure?
• Modification—What evolving technologies, opportunities, and situations might require future modifications to modules?

Reactive Domains:
• Correction — What types of response activities might fail and need correction?
• Variation — What operational conditions and resources vary over what range when response capabilities must be assembled?
• Expansion/Contraction — What are the upper and lower bounds of response capacity needs?
• Reconfiguration — What types of situations will require modular system reconfiguration to respond effectively?

Concluding Remarks

So called fourth-generation warfare is characterized as relatively small guerilla group activity, and fifth-generation warfare is characterized as super-empowered individuals. Both leverage newly affordable technology to nonconventional advantage—and though the term warfare conjures up attacks on nation-states by other nation-states, it also encompasses undesirable intervention directed at any entity, from a concerted effort to gain advantage on a commercial competitor to an individual’s revenge against a perceived injustice. But affordability is relative. With high stakes, the resources applied by organized crime and nation-states are less constrained, with access to the brightest of minds and the most sophisticated technologies.

Fielding sustainably secure systems today is critical to system mission needs, yet difficult when system security is less than a paramount thoughtful concern of the system engineering processes. Responsibility lies with both the acquirer, to demand it, and the supplier, to provide it even when not demanded. The acquirer must place responsibility for system security on the systems engineering activity. The supplier must enable sustainable security and enable agile lifecycle security processes throughout the operational lifetime.

References

Security engineers implicitly or explicitly develop two types of systems models. One type of model captures the operational end state of the security of the system and gives us useful data on the security performance within the finished system. The other type helps us enable, control, and measure the development of the system. In this article, the term model refers to an abstraction created to represent reality (such as simulations and performance metric models) and to represent a standard or best practice (like an organization’s standard model for lifecycle development). In this article, I will attempt to briefly describe some of these models used by security engineers, provide some examples, and (perhaps more importantly) point you to where more information is available.

Lifecycle Models

As systems engineers, we are experienced with many models of the system-development lifecycle (SDLC). We define our activities using the phases of systems as they transition from concept to retirement. Through our membership in INCOSE, we have access to versions of the SDLC that can be tailored to virtually any systems engineering application. For program planning, these lifecycles can be decomposed into the six often cyclical stages in figure 1 (the cycles are shown by feedback lines).

These six stages may be understood as follows:

- **Definition.** Activities include learning the operational environment (the mission area), capturing the mission essentials and mission needs in some form of document (e.g., an initial capabilities document or mission needs statement), and defining the desired outcome of the object of engineering (for systems, this is usually accomplished by creation of a high-level requirement specification and a concept-of-operations document).
- **Design.** In “design” the concepts from the definition phase are modeled and translated into artifacts that enable and constrain the creation (engineering) of the object. This stage includes the creation of the conceptual models (e.g., architectural views) for the system.
- **Development.** During this stage, the objects of engineering are created.
- **Deployment.** In some systems engineering lifecycle models, this stage is called integration, verification, and validation. It is the phase where the engineering objects are integrated in large systems, requirements are verified, and the completed system is validated to ensure it fulfills the need captured during the definition phase.
- **Operations.** The system is operated and sustained. This phase contains engineering activities that include maintenance, periodic upgrade, and (potentially anticipated) repair of latent defects.
- **Retirement.** This will include migration of the activities supported by the existing system to new systems, disposal of the existing system, and capture of intellectual property (e.g., lessons learned).

System security engineering is a specialized subfield of systems engineering that focuses on the security aspects in the engineering of systems (primarily information-technology systems). As is the case in systems engineering, many organizations provide frameworks, standards, and guidance for security engineering. For systems funded by the United States government, the framework established by the US National Institute of Standards and Technology (NIST) to provide guidance for development and maintenance of the risk management program is the Risk Management Framework (RMF; NIST 2011). All federal systems (including US Department of Defense and the US intelligence community) currently base their security-approval processes on this framework, or are in the process of doing so.
According to this US standard, security engineering is performed through the following six phases.

1. **Categorize Information System.** In this phase, security categorization is conducted using US Federal Information Processing Standards Publication 199 (FIPS 199) titled *Standards for Categorization of Federal Information and Information Systems* (FIPS 2004). NIST provides the special *Guide for Mapping Types of Information and Information Systems to Security Categories* (NIST 2008) for additional guidance on how to conduct this categorization.

2. **Select Security Controls.** Common controls are security controls that are inherited by the system under development from the organization owning the system. The organization identifies these common controls using FIPS 200 *Minimum Security Requirements for Federal Information and Information Systems* (FIPS 2008) to ensure that the security capability provided by the inherited controls is consistent with other existing or planned components. These “common controls” are then used to help select the security controls applicable to the system under development. If deemed necessary after a security-risk assessment, additional requirements may be added to supplement this set of common controls. The NIST publication *Recommended Security Controls for Federal Information Systems and Organizations* (NIST 2009) contains the baseline set of controls.

3. **Implement Security Controls.** As the system matures, the security architecture is developed to support the allocation of the security controls from phase 2.

4. **Assess Security Controls.** Security-control assessments are evaluations conducted to identify potential weaknesses (deficiencies) in system early in an effort to provide the most cost-effective method for initiating corrective actions.

5. **Authorize Information System.** Although the ultimate goal of this phase is the authorization from the organization to operate a system, this phase also includes the steps that lead up to this “authorization decision” and, if needed, may include any liens (usually requiring a “get-well plan”) that must be addressed for an approved system to continue to be allowed to operate.

6. **Monitor Security Controls.** Often called the continuous monitoring phase, this phase coincides with the operations of the system. During this phase, the security controls, and any other metrics determined to be of interest, are monitored with the intent of identifying any security relevant changes that may require the re-evaluation of the authorization to operate the system.

When we compare the six steps of our example systems engineering lifecycle model with the six steps of the Risk Management Framework, there at first appears to be almost one-to-one comparison. Both have an iterative series of phases that grow in detail as the development matures, but further analysis highlights some perturbations (although the relationships may differ depending on an organization’s implementations of these model). In general, system-requirements definition (at least in draft) is needed for to complete the Categorization phase and the Select Security Control phase (basically, definition drives categorization and selection). Categorization and selection must be complete (at least in draft) to complete Design. Since the initiation of the implementing security controls phase must have (as a minimum) a systems design, these two phases roughly align. Again, though, systems design will slightly precede implementing security controls. Implementing Security Controls and Assessing Security Controls roughly align respectively with Development and Deployment. In practice however, the implement and assess activities of the RMF are iterative and occur multiple times throughout the time period contained within the Development and Deployment phases. Authorization aligns roughly with the Deployment, as does Monitor with Operations. There is no matching RMF phase for the Retirement Lifecycle phase. Figure 2 graphically shows this rough comparison.

![Figure 2. The relationship of the SDLC to the RMF](image-url)

Another lifecycle framework sometimes referenced by security engineers is the Information Assurance Technology Framework (NSA 2002). The RMF is an evolution of the principals of the Information Assurance Technology Framework, so the RMF is most often used for authorization activities; the Information Assurance Technology Framework however, is rich with excellent security-process definitions and is an excellent source of guidance for systems and security engineers.

**Process Model**

As systems engineers, we use process models to add discipline to systems we help develop. For example, we often use the CMMI (see [http://cmmiinstitute.com/](http://cmmiinstitute.com/)). Through our INCOSE association, we have a thick and rich resource pool from which to draw process model guidance and examples. For the security engineer,
the process models are a little harder to find and process descriptions have often been very organizationally dependent. The most internationally recognized capability model for system security engineering is the Information Technology-Systems Security Engineering-Capability Maturity Model SSE-CMM (ISO and IEC 2002). This capability maturity model was originally created as an initiative of the US National Security Agency; a very useful repository of capability model documentation (and history) is available at http://www.sse-cmm.org/.

Baseline Models of the System

Security engineers define the security controls (security requirements) that drive the implementation of the necessary safeguards and countermeasures for a system to minimize or mitigate security risks. Although there are several taxonomies (catalogs) of security controls available that can be used by the security engineer, to comply with US federal standards however, the security engineer must use the catalog of controls defined in NIST SP 800-53. The baseline set of controls must next be tailored to provide a system-specific set of controls that will be used for implementation, assessment, authorization, and continuous monitoring.

To accomplish this tailoring, security engineers perform detailed security-risk analyses and assessments. For US federal systems, the guidance is provided in NIST SP800-30 (NIST 2012).

There are many sources of practical guidance and supports tools for doing risk assessments. For example, see the OCTAVE tool developed by the Software Engineering Institute of Carnegie-Mellon University, Pittsburgh, Pennsylvania, US, at http://www.cert.org/octave/. Nevertheless, most methods use some form of attack-tree analysis. Attack trees are multileveled diagrams that start with one root and then branch out into layers of parents and children. Starting with the root (the highest-level parent), any parent may have multiple children, but all children have only one parent. From the bottom up, child nodes are conditions which must be satisfied to make the direct parent node true; when the root is satisfied, the attack is complete. Each node may be satisfied only by its direct child nodes. The security engineer will determine the conditions necessary for a risk to be realized (these become the child nodes) and then by associating a probability with each child and following the logic rules associated with the attack tree structure, the security engineer can estimate a risk value for each risk.

These early models based on attack-tree structures, which are used to evaluate the potential risk associated with a system, are the first and best estimate of how secure a system is using a reasonable monetized estimate. Attack-tree-based models also allow the security engineer to normalize each alternative, helping to optimize alternative analysis. This monetized set of estimates has the added benefit of providing an estimate (albeit a qualitative one) of the unmitigated risk, the mitigated risk, and the cost associated with each mitigation that can (if necessary) be used to support the overall program or system’s risk-management process. Once complete with risk analysis and analysis of alternatives, the final tailored security controls are captured in formal documentation (usually some form of system security plan or system security requirements-traceability matrix) and become the model used by the system stakeholders as the permanent baseline of the completed system. This model is the basis for the Assess, Authorize, and Monitor phases of the RMF.

Conclusion

As security engineering continues to mature, research and development of security engineering models will also mature. The recent acknowledgement of security engineering as a discipline within systems engineering will increase the pace of the development of these models. Security engineering will be a long valuable and growing part of the Systems Engineering Handbook. Look to the Systems Engineering Handbook in the near future as a valuable and persistent source of guidance in how to employ security engineering.

References


Mission Thread Security Analysis: A Tool for Systems Engineers to Characterize Operational Security Behavior

Carol Woody, cwoody@cert.org

Today’s software is riddled with weaknesses and vulnerabilities that represent unacceptable security risks because the weaknesses allow attackers to gain system access and bypass security controls (McGraw 2006, 3). Why should systems engineers be concerned about this? Software is built or acquired to meet requirements established by systems engineers. The percentage of system functions performed by software has risen exponentially in recent years. A 2010 report by the Computer Science and Telecommunications Board noted, “Software has become essential to all aspects of military system capabilities and operations.” In 1960 software handled 8% of the F-4 Phantom fighter’s functionality; it expanded to 45% of the F-16 Fighting Falcon in 1982 and to 80% of the F-22 Raptor in 2000 (CSTB 2010, 19). Military systems are not the only ones affected. Software controls virtually everything we use these days; cars, planes, banks, restaurants, stores, telephones, appliances, and entertainment devices rely extensively on software. Organizations in just about every field rely on software even though it contains critical weaknesses that represent serious security risks. This article describes the use of mission thread security analysis to provide systems engineers with a tool that connects desired system functionality with the underlying software to evaluate the sufficiency of requirements for software security.

A mission thread is an end-to-end set of steps that illustrate a system’s expected behavior under a set of conditions and provides a basis for identifying and analyzing potential security risks.

Interconnectivity of systems has increased along with the reliance on software. Operational missions (and business processes) rely on multiple systems working together to address a required capability. Systems engineers must carefully analyze expected functional use within and across multiple systems to identify weaknesses that will allow attackers to bypass security controls and to build in appropriate mitigations to block attackers.

Another key challenge for systems engineers in addressing security is the expanded reliance on commercially available software components. Requirements must be sufficiently complete to support selecting products for components that address needed functionality. In addition, requirements must ensure that no one can use the component to jeopardize mission success. The requirements must include consideration of software weaknesses within these products and protection mechanisms to limit risk to the operational mission. Systems engineers can use mission threads to analyze security and increase confidence in the completeness of security requirements for both acquisition and development.

Mission Thread Analysis

A mission thread is an end-to-end set of steps that illustrate a system’s expected behavior under a set of conditions and provides a basis for identifying and analyzing potential security risks. For each mission step, systems engineers characterize the underlying software components as well as technology and human interactions. Systems engineers already use parts of this analysis. They develop operational scenarios to support a conceptual design. However, their content is typically focused on how stakeholders would like the system to operate without considering how the components, if built as designed, will actually work (and possibly fail) together.

Mission thread analysis can help systems engineers establish how a system contributes to mission success and to evaluate the ways, intentional or unintentional, that system failures could occur and how these would impact the mission. This analysis should start...
early in the lifecycle. By evaluating the composition in addition to the components, systems engineers can validate that the composition as conceived, then designed, and finally built will perform with expected security. Confirmation that the components appropriately respond to expected operational use increases confidence that the system will function as intended even in the event of an attack (Ellison et al. 2008). The simplest way to show the value of this approach is through an example. I have selected the Commercial Mobile Alert System (CMAS) to demonstrate how to use a mission thread to analyze security.

The Commercial Mobile Alert System as an Example of Mission Thread Security Analysis

The Commercial Mobile Alert System enables local, tribal, state, territorial, and federal public-safety officials to send geographically targeted text alerts in the United States for public distribution by mobile carriers (FEMA 2012). The US Department of Homeland Security Science and Technology Directorate partnered with the Federal Emergency Management Agency (FEMA), the Federal Communication Commission (FCC), and commercial mobile service providers (CMSPs) to incorporate wireless capability into FEMA’s Integrated Public Alert and Warning System (IPAWS). Customers who own CMAS-capable mobile phones will automatically receive these alerts during an emergency if they are located in the affected geographic area. Public-safety officials and alert recipients will rely on CMAS capabilities if they have confidence that the alerts are accurate and timely. Effective security is required to support this confidence. Attackers could create false alerts or cause valid alerts to be delayed, destroyed, or modified. This would place the alert-originating organization’s mission, and the lives and property of the citizens it serves, at risk.

CMAS Mission Thread

Begin with the development of an operational mission thread. For CMAS, 25 steps typically take place from the determination of the need for an alert to the receipt by cell-phone owners:

1. First responder contacts local alerting authority via an approved device (cell phone, e-mail, radio, etc.) to state that an event meets criteria for using CMAS to issue, cancel, or update an alert and provides information for message.
2. Local alerting authority (person) determines that call or e-mail from first responder is legitimate.
3. Local alerting authority instructs Alert Origination System (AOS) operator to issue, cancel, or update an alert using information provided by first responder.
4. AOS operator logs on to the AOS.
5. AOS log-on process activates auditing of the operator’s session.
6. AOS operator enters alert, cancel, or update message.
7. AOS converts message to a format compliant with the Common Alerting Protocol (CAP, a CMAS input standard).
8. CAP-compliant message is signed by a second person for local confirmation.
10. IPAWS-OPEN Gateway verifies message and returns status message to AOS.
11. AOS operator reads status message and responds as needed.
12. If the message was verified, IPAWS-OPEN Gateway sends message to CMAS Alert Aggregator.
13. CMAS Alert Aggregator verifies message and returns status to IPAWS-OPEN Gateway.
14. IPAWS-OPEN Gateway processes status and responds as needed.
15. CMAS Alert Aggregator performs additional message processing as needed.
16. If the message was verified, CMAS Alert Aggregator transmits alert to Federal Alert Gateway.
17. Federal Alert Gateway verifies message and returns status to CMAS Alert Aggregator.
18. CMAS Alert Aggregator processes status and responds as needed.
19. If the message was verified, Federal Alert Gateway converts message to CMAC (Commercial Mobile Alert for Interface C) format.
20. Federal Alert Gateway transmits message to CMSP gateway.
21. CMSP Gateway returns status to Federal Alert Gateway.
22. Federal Alert Gateway processes status and responds as needed.
23. CMSP Gateway sends message to CMSP Infrastructure.
24. CMSP Infrastructure sends message via broadcast to mobile devices in the designated areas.
25. Mobile device users (recipients) receive the message.

Note that although many of the steps do not involve technology, they can still represent security risks to the mission. Security analysis must take into account the people and their interactions with technology in addition to the functioning of a system itself.

Most security evaluations consider only internal system execution. However, the mission thread must cross organizational and system boundaries to be complete. Mission thread security analysis provides a way of confirming that each participating system is secure and does not represent a risk to all others involved in mission execution. Figure 1 provides a picture of the CMAS mission thread and includes step numbers from the list to link each step to the appropriate system area.
Successful completion requires flawless execution of four major system areas—alert originator, FEMA IPAWS system, CMSPs, and cell phone recipients—each shown in a row of the figure. Each area operates independently, connected only through the transmission of a CMAS alert. The interactions (not the decomposition) are critical to mission success.

**CMAS Mission Thread Security Analysis**

Using the operational flow illustrated in figure 1, next identify potential security threats. In the CMAS example, we used the STRIDE Threat Method for the threat evaluation. STRIDE, developed by Microsoft, considers six typical categories of security concerns: spoofing, tampering with data, repudiation, information disclosure, denial of service, and elevation of privilege (MSDN 2005; Howard and Lipner 2006). We will focus on steps 4–9 of the mission thread, which represent the transition across two major system areas from the alert originator to the FEMA system and provide an opportunity for mission failure if interaction between the system areas is not secure. Table 1 shows the result of the STRIDE analysis on the selected steps. For each step, we analyzed the technology assets critical to step execution to determine ways that STRIDE threats can compromise each. We

---

**Table 1. STRIDE analysis for selected CMAS mission thread steps**

<table>
<thead>
<tr>
<th>Step No.</th>
<th>Step Description</th>
<th>Assets</th>
<th>STRIDE Threat Identification Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>AOS operator attempts to log on to the alert origination system.</td>
<td>One person Server (valid accounts/ authentication information) Log-on procedure Log-on application Username/password data in database Communications between log-on software, server, and AOS</td>
<td>S: Unidentified individual attempts to log-on with AOS operator’s information R: AOS operator denies having logged on I: Capture of log-on info using key logger or packet sniffer D: AOS operator’s account not registered or servers are down E: Successful log-on by an unidentified and unauthorized individual</td>
</tr>
<tr>
<td>5</td>
<td>AOS log-on activates auditing of the operator’s session.</td>
<td>Auditing application Auditing procedure Communications from accounts to auditing application Local or remote storage</td>
<td>T: Logged entries added, deleted, or modified inappropriately I: Logged entries containing credential data are compromised D: Log full or server unavailable</td>
</tr>
<tr>
<td>6</td>
<td>AOS operator enters alert, cancel, or update message</td>
<td>One person Alert scripts Procedures for building scripts Graphical-user-interface application Communications between application and alert-generation software (including server and application)</td>
<td>T: Formatting errors produce incorrect message D: Scripts are unavailable or corrupted</td>
</tr>
<tr>
<td>7</td>
<td>AOS converts message to CAP-compliant format required by IPAWS.</td>
<td>Conversion application</td>
<td>T: Data is changed between the AOS and the server D: Server is down</td>
</tr>
<tr>
<td>8</td>
<td>CAP-compliant message is signed by two people.</td>
<td>Signature entry application Signature validation application Public/private key pair for every user</td>
<td>S: Digital signature is falsified R: User claims not to have signed D: Server goes down so keys cannot be distributed, or keys have expired and message cannot be sent</td>
</tr>
<tr>
<td>9</td>
<td>AOS transmits message to the IPAWS-OPEN Gateway.</td>
<td>Application that securely connects to IPAWS Information used to authenticate AOS and IPAWS</td>
<td>S: Falsified AOS CAP message or IPAWS gateway attacked and site is redirected T: Data within message is modified I: Message is not encrypted and credentials are visible D: IPAWS-OPEN Gateway is down</td>
</tr>
</tbody>
</table>
selected threats appropriate to the assets used in that step (Howard and Lipner 2006, 119). Security and software experts need to participate in this portion of the analysis to bring their expertise of what can go wrong and the potential impact of each possible failure into the analysis.

Next, we interviewed alert originators and vendors of alert-origination systems to confirm the validity of the information assembled about the selected mission steps and to gain further insight into ways in which security issues might arise. We also reviewed available documentation from the US Department of Homeland Security on expected AOS performance and IPAWS capability.

Based on this input, security experts identified two security risks that could lead to mission failure:
1. Authentication of the individual using the AOS in step 4;
2. Validation and protection of the digital signatures applied to the alert approved for submission to the Alert Aggregator in step 8.

Systems engineers reviewed the requirements to identify additions needed to reduce the likelihood of these risks. Systems engineers using a traditional analysis approach might identify and address the first risk by considering “least privilege” concerns (Saltzer and Schroeder 1974) for the structure of roles in system usage. However, the second risk is tied to the interchange between two systems (AOS and IPAWS) and falls outside the direct control of either system. Mission thread security analysis provides a framework in which to reason about this type of security failure.

**Conclusion**

Analyzing the security of the CMAS mission thread provided a mechanism to identify security issues that would have been missed by typical systems engineering methods. The process of developing a well-articulated mission thread that systems engineers, software engineers, and security experts can share and analyze provides an opportunity to uncover missing or incomplete requirements as well as differences in understanding, faulty assumptions, and interactions across system and software boundaries that could contribute to security concerns and potential failure (Ellison et al. 2008).

The mission thread analysis connects each mission step with the technology and human assets needed to execute that step and provides a framework to link potential security threats directly to mission execution. Mission thread diagrams and tables assemble information in a structure that can be readily reviewed and validated by systems engineers, security experts, system users, and technology experts from the various disciplines involved in acquisition, development, and operational support. Mission thread security analysis can provide systems engineers with an effective tool for improved identification of security and software risks to increase confidence that the system and software will function with appropriate operational security.

**References**


Careful integration of security functions and interfaces is critical to any successful information system security engineering effort. The process for information system security integration is very similar to that used for traditional systems engineering when applied to information technology. Due to the complexity of modern information systems and information system interfaces as well as the “intangible” nature of purely software products, failure to properly integrate system security functions across components or with external systems can lead to significant security flaws that may be very difficult to detect. Additionally, while there are numerous interface standards available for creation of product interfaces, many commercial products will implement the standards with added features or capability that may or may not function properly across different vendor product offerings.

Careful application of the system engineering integration principles to information technology using methods customized to support information system security engineering will alleviate potential security flaws introduced through the interfaces of integrated software and hardware products. The security engineer is concerned with all system elements and interfaces as they relate to the concepts of confidentiality, integrity, and availability. Application of the confidentiality, integrity, and availability concepts is within the purview of the security engineer but does overlap heavily with system functional requirements in many cases.

Inheritance of Security Functions

A key concept generally known in the information system security field as “inheritance” is, in systems engineering terms, simply the allocation and integration of security related functions or requirements between systems within a larger environment. The term inheritance is most typically used in reference to information system certification and accreditation, such as the DoD Information Assurance Certification and Accreditation Program, or other system authorization processes, such as the NIST Risk Management Framework. Inheritance illustrates a key problem in system security engineering within the information technology framework, which is that any one system or component simply cannot accomplish all the required security functions within the system boundaries. While there are some exceptions for non-networked technology, most common information technology systems connected to larger networks rely on external systems for a portion of their security functions, which means some integration effort is generally required to ensure the required security functions are being correctly and effectively “inherited” by a system or component receiving the security function from another system or component.

In many ways the close interdependency on security functions constitutes a fundamental difference between systems engineering in large physical systems (e.g., a tank or ship) and systems engineering in information technology systems. Most physical systems and major subcomponents have the ability to perform their design function as a self-contained unit at least part of the time. Often, they can be individually tested or validated at the component level prior to final assembly. However, with most networked information technology systems there is a continuous reliance on security providers external to the system during all phases of the system operation. It is often even difficult to fully test some security functions until the final product is introduced into the operational environment and infrastructure related security can be “inherited.” In fact, many of the “inherited” security functions may have no direct system-to-system interface between the protected and protecting systems. Where traditional systems engineering would develop interface control documents to define the interface, this is often impractical in security engineering when many components may not share direct interfaces.

There is often some level of integration effort between the protected system and the external security provider providing inherited security functions, even though there may be no direct system-to-system interface. This may range from a basic coordination effort to formal configuration management of security settings on external systems.
Network Devices and “Interfaces”

In the formal systems engineering integration process, component integration is largely based around the definition of component values (size, weight) and control of the component interfaces with other components or systems. In the realm of information technology existing within complex network infrastructure, it can be problematic to define what external systems or components have “interfaces” with a target system. Most networked systems inherit (receive) security functions from external systems or components without necessarily having a direct interface with those other systems or components. As an example, a common security function inherited by most network systems is protection by a network firewall device somewhere between the protected system and public networks like the Internet. While the protected system may not have any direct data or information exchange with the firewall device, the system may have one or more logical (e.g., TCP/IP) interfaces with external systems (outside the firewall) that transit across the firewall device. Normally these interfaces transit the firewall transparently to the protected system but changes to the firewall configuration could adversely affect the interface.

Does this mean that the firewall’s relationship to the protected system should be treated like an interface in systems engineering? While it would be possible to do so, this can be problematic in many instances due to the very large number of systems typically protected by a firewall. The firewall may also have well in excess of 50,000 individual configurations (e.g., firewall rules), any of which could potentially affect interfaces crossing the device. Also, in many cases the firewall may not be controlled by the same organizational entity as either the protected system or an external system with which it interfaces across a firewall protected boundary. To further complicate this scenario, there would typically be a number of network protective devices besides just the firewall along the communication path that provide protections when properly configured but also introduce the risk of interface failure if misconfigured. While system or component interfaces that actually involve the exchange of information can be defined in the traditional sense, many other “interfaces” involving security functions may need to be defined using something other than a traditional interface control document.

Coordination with Other Information Assurance Processes

In order to comply with the United States Federal Information Systems Management Act (FISMA), US federal agencies are required to authorize systems prior to those systems being brought online in accordance with one of the approved certification-and-accreditation or system-authorization processes. Other countries use comparable processes or practices. The primary process used in the United States is the National Institute of Standards and Technology (NIST) Risk Management Framework. The US Department of Defense uses the Defense Information Assurance Certification and Accreditation Program (DIACAP) which has a number of similarities to the NIST process. Both of these processes have explicit requirements for the documentation of interfaces between systems, although the exact format and content of the interface documentation does vary. The level of information required in interface documentation is typically more administrative in nature than that referenced by the INCOSE handbook for a formal systems engineering interface control document; however there is significant overlap in these requirements.

The systems engineering documentation developed during system integration process, primarily the interface control documents, can and should be used as a supporting resource for the development and support of the certification-and-accreditation or system-authorization processes. If properly constructed, the system integration documentation will fulfill the mandatory interface-documentation requirements for most federal authorization or accreditation requirements as well as providing supporting artifacts for a number of other mandatory federal information-assurance controls (e.g., transmission security requirements). Additionally, much of the testing (verification and validation) required within the certification-and-accreditation processes can be leveraged as part of the overall system testing program.

In addition to directly supporting federal information-assurance requirements, the systems engineering interface control document may also be used to support other similar system-authorization or accreditation processes, many required by federal law, regulation, or industry standards. For instance, the US Health Insurance Portability and Accountability Act (HIPAA), the global Payment Card Industry Data Security Standard (PCI-DSS), the US Federal Education Right to Privacy Act (FERPA), and the ISO/IEC 27000 family of standards all contain requirements or elements that can be supported by the systems engineering documentation produced as part of the systems-integration process. This includes the internal system architectures and interfaces as well as the interface control documents for external interfaces. When the parallel information-assurance and security-engineering processes are considered for a particular system, significant cost savings can be achieved through the production of systems engineering system-integration documentation that is consistent with and supportive of the information requirements in the security process.

Example system

Figure 1 shows a notional network architecture and system. The network architecture consists of the internal network, the external network, and two specially configured perimeter network segments, commonly referred to as a DMZ segment, which uses a firewall device to separate the internal and the external network. The four system components are: (1) an internal database server, (2) a communication proxy in a protected DMZ, (3) a publicly accessible web server in a protected DMZ,
and (4) a system-administration subsystem in the internal network. The notional system has a requirement to interface with an authentication system, a backup system, a messaging system, internal users, and an external commerce portal. This is a realistic configuration for a simple multicomponent data system utilizing security services from other organizational systems. It also facilitates passing certain logical interfaces through the firewall device, communication proxy, or through an externally facing web-server component for external communications interfaces. However, there are several areas in this notional design where the system-integration process is crucial for resolving potential issues and developing appropriate documentation to support the system-security state and functionality. The three areas of discussion for this example will be interfaces occurring within internal network segment, interfaces between system components on the internal network segment and the DMZ segments, and interfaces between the notional system and the external commerce server. For purposes of the example, we will assume all the data that requires protection resides exclusively on the database server and other three components facilitate access to the data by various means.

For the interfaces and integration effort between the notional system and the internal interfacing systems residing on the internal network, the interface descriptions will likely be fairly standard and require data elements consistent with a typical systems engineering interface control document. These interfaces are likely to utilize either open or proprietary data-connection standards for the authentication, backup, and messaging functions. However, functional integration efforts from the security-engineering perspective may have to go beyond that which would be required for a purely functional integration to ensure data protection requirements are met. For instance, the functional requirements for the interface to the backup system may simply require a data-transmission path and proper functioning of the backup system with respect to the example system. From the security perspective additional considerations may include the encryption of the data over the internal network and the encryption of data within the backup system. Depending on the sensitivity of the data and the applicable laws and regulations, a detailed analysis of the encryption mechanisms within the backup system may also be required. This will often require enforcement of particular configuration settings on both the example system and the backup system.

Additional data-protection controls designed to prevent inappropriate leakage of sensitive data may need to be applied to the interface with the messaging system. This may require specific design configurations within the example system during the integration process to ensure that protected information cannot be accidently or maliciously released through an interface to the messaging system. Depending on the authentication system, at time of connection during the integration process, policy and system functions associated with the authentication system may cause unexpected behavior or failures that will need to be resolved. If not fully understood, this can further complicate the integration due to the authentication system, potentially forcing changes upon the example system components. A specific example of this behavior would be the connection of the example system to Windows Active Directory where mandatory group policy is applied to member systems that affect component behaviors by overriding any prior configuration settings. In many cases the effects of group policy settings may not be realized until the initial system connection during the integration pro-
Security engineering during the integration process is critical to ensure proper application and documentation of the security controls or requirements. Many security controls cannot be fully applied until a system is integrated into an operational environment and interfaces are made between the system and the operational instances of other systems that applied protective services or security services to the system. Additionally, the documentation provided by the systems engineering integration efforts will facilitate security-related approvals or authorizations to operate and will reduce costs to produce documentation to support system security authorization. Failure to fully consider the security engineering elements applied to a system during integration will significantly increase the likelihood of failures during final operational integration. This is especially true for network-connected systems.

References
Verifying Security-Control Requirements and Validating their Effectiveness

Bruce Hunter, bruce.hunter@incose.org

A
though security aspects of a system may come from multiple sources (customer requirements, threat and vulnerability analysis, or security incidents), they must still be verified and validated like any other requirements as part of a systems approach. This article shares experience and practical guidance on the verification and validation of required security controls that must be performed along with other functional, nonfunctional and operational system requirements if the system is to securely meet (and continue to meet) its mission. While this article focuses on information systems, its advice is equally applicable to other security engineering activities and to controls (logical or physical), and is not specific to any technology solution.

**Security Context**

Verifying and validating system-security requirements needs a different approach from other system requirements due to a limited knowledge of the extent of security threats and vulnerabilities at the time of specification and their evolution through the development and support lifecycles.

**Security Model**

One needs an understanding of a generic security model and how security controls (countermeasures) protect assets from exploitation of vulnerabilities to successfully plan and perform system security validation and verification (V&V). The security model (ISO/IEC 15408:2009) in figure 1 provides a good guide for the interaction between probable threats and vulnerabilities. Additions to the model are in red and show how verification and validation confirm that countermeasures (controls) securely protect the systems assets.

From a systems engineering perspective, the information security model provides the basis for understanding and verifying the controls required to protect the confidentiality, integrity, availability, and traceability of assets of the system.

**Security Assets**

Assets protected by security controls or countermeasures are typically:

- Information—static or in transit
- Functionality—performance or reliability
- Monetary—integrity
- Safety—failure to operate or false operation

The required countermeasures or controls to protect these assets become the security policies or requirements.

**Security Verification and Validation Approach**

Verifying and validating system security requirements has some unique aspects that differ from system functional and performance test. For one, it is similar to system safety methodology (Hunter 2009): both share a common emphasis on system and human failures, not on functional requirement implementation. Verifying and validating system security requirements is also prone to conflicting requirements and implementation where the...
functionality introduced may increase vulnerability or defeat established security controls. Verifying the effectiveness of controls has an emphasis on testing-to-fail rather than the emphasis on testing-to-pass with functional testing.

Finally, this type of verification and validation is difficult to assess for sufficient coverage in planning or performance, and relies heavily on judgement by subject-matter experts.

Planning for Security Verification and Validation
Planning system security verification and validation must take into consideration not only the necessary resources, tools, skills, and schedule, but also the impact such testing may expose through publication of any weakness found or critical control information that may lead to its exploitation.

Security Verification Objectives
Concentration on the objectives of security requirements confirms protection of the system assets is effective against probable threat scenarios affecting the following:

- Confidentiality (confirm sensitive or private information is protected from unauthorised disclosure)
- Integrity (confirm the accuracy, completeness and validity of the asset is protected from exploitation)
- Availability (confirm the asset is reliably ready for use when needed)
- Traceability (confirm any interaction with the asset can be traceable to a specific authorised party; establish forensic information allowing investigation of any unauthorised actions; and provide evidence of a chain of custody)

Each of these objectives must be accounted for in verification and validation planning to prove security requirements.

Security Verification and Validation Methods
The verification (requirements have been met) and validation (implemented requirements are effective) methods and practices for security testing differs from those proving functional requirements but can be categorised into the same activities as shown in the following table with typical security methodology examples.

Validation of security controls can include interviewing of associated staff for security awareness to confirm that vulnerabilities by human interactions are not being introduced, as in social engineering or phishing. Standards provide a good set of security tests and assessments (NIST SP800-115 2008; ISO/IEC 15408:2009) that are applied.

<table>
<thead>
<tr>
<th>Security Aspect</th>
<th>Verification Approach</th>
<th>Validation Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation Review</td>
<td>Controls have been incorporated in design and support documents</td>
<td></td>
</tr>
<tr>
<td>Log Review</td>
<td></td>
<td>Scan for suspicious behaviour and confirm that access rules have been obeyed and enforced.</td>
</tr>
<tr>
<td>Rule Set Review</td>
<td>Verify that firewall rules have been set and maintained in accordance with policies and known vulnerabilities.</td>
<td>Validate that only authorised configuration changes have been made.</td>
</tr>
<tr>
<td>Security Configuration and Port Identification</td>
<td>Verify that system configuration has been set as per requirements including management of firewall, switch, and malware and intrusion protection systems.</td>
<td></td>
</tr>
<tr>
<td>Software Development Lifecycle and Security Standards</td>
<td>Verify that applicable security design and coding techniques have been employed. Verify independent checking and code inspection has been employed to prevent accidental or intentional vulnerabilities.</td>
<td></td>
</tr>
<tr>
<td>Network Sniffing, Network Discovery, Penetration Testing, Brute Force Testing and File Integrity Checking</td>
<td></td>
<td>Confirm protected network elements cannot be accessed or breached</td>
</tr>
<tr>
<td>Standards Compliance</td>
<td>Verify that applicable requirements from standards have been applied</td>
<td>Validate that system has been hardened against known vulnerabilities.</td>
</tr>
<tr>
<td>Vulnerability Scanning</td>
<td></td>
<td>Validate that wireless access points have not been inadvertently enabled</td>
</tr>
<tr>
<td>Wireless Scanning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web vulnerability (e.g., XSS, SQL Injection)</td>
<td>Validate that known exploitation methods do not allow privileged access to databases and other web assets</td>
<td></td>
</tr>
<tr>
<td>Password Cracking</td>
<td>Verify that default passwords have been removed</td>
<td>Validate that password-cracking attempts on access are detected, delayed, and alerted.</td>
</tr>
<tr>
<td>Least Privilege Access</td>
<td>Verify that identity and access management has been implemented so that users have least privilege.</td>
<td></td>
</tr>
<tr>
<td>Social Engineering</td>
<td>Verify security awareness initiatives have been deployed to all users.</td>
<td>Validate that social engineering attempts are changed</td>
</tr>
<tr>
<td>Unauthorised Remote Access</td>
<td>Validate that remote access attempts with possible exploitations are not successful</td>
<td></td>
</tr>
<tr>
<td>Application Security Testing</td>
<td>Verify that application vulnerabilities have been identified and protected measures have been employed</td>
<td>Validated that added applications do not degrade existing security controls</td>
</tr>
<tr>
<td>Patch Management</td>
<td>Verify that patch management practices are in place</td>
<td>Validate that latest patches have been fully deployed in accordance with policies</td>
</tr>
</tbody>
</table>
A well-founded checklist for evaluation of security-policy implementation, identities evidence of supporting documentation, and practices exhibiting operational effectiveness. Test procedures benefit from periodic revalidation of the controls and incorporation of new control validation.

Test Scope
The scope of security testing should meet the identified system security requirements, and also should be extended to include any path that a threat may exploit. Testing the system-security scope for threat vectors includes the following:

- Perimeter testing:
  - Penetration Testing
  - Denial of service testing
  - Port identification
- Identity and access management testing:
  - Role-based access
  - Least privilege access
- Computer hardening:
  - Verification that insecure features are disabled
  - Default BIOS/OS access codes are changed
  - Elimination of back door
- Typical threat vectors include these:
  - Network connections
  - Unused ports such as Wi-Fi and Bluetooth
  - Physical security of unencrypted network sections
  - User accessible equipment ports
  - Unsecured USB ports allowing malware injection inside the security perimeter (e.g., Conficker, Stuxnet)
- System operating, maintenance, and support staff
  - (It may seem surprising, but either malicious insider threats or accidentally introduced vulnerabilities by insiders are a large part of successful security breaches.)
- Web based threats:
  - Cross-site scripting
  - SQL Injection

The issue presented to the contemporary security tester is that these threat vectors keep changing. To avoid this, review test scopes against current threats and vulnerability from CERT organisations or security product vendors (for a good example, see McAfee Labs 2012).

Security-Testing Resources
Security-testing techniques do require experienced and trusted subject-matter experts to ensure that testing finds any weakness, and to ensure that at the same time the testing does not introduce new vulnerabilities in the system, even for a brief period. Engagement of refereed testers early is essential to make sure achievement of the plan, as this may identify new security requirements that can be fed into the development lifecycle. Building security in is more effective than trying to test vulnerabilities out.

Be careful that test resources can be trusted. The use of “Black Hats” would certainly provide rigour and persistence in uncovering vulnerabilities, but privileged system profiling and introduction of vulnerabilities may not be worth the risk. Understanding how threat agents such as “Black Hats” do exploit system vulnerability is desirable in setting up test scenarios. Books such as Kingpin are a recommended reading (Poulsen 2011) to understand the mindset of a cyber criminal and the tricks they employ.

Performing Security Testing

V&V Readiness
Readiness criteria for security testing include the following:

- Confidentiality agreements are established for test participants to ensure vulnerabilities are not publicised before correction or not divulged allowing future exploitation.
- Security subject-matter experts are engaged to conduct testing and analyses.
- Test environment and test data is available that will not expose a live system to secure data loss.
- Sets of desensitised test data that can be compromised if necessary.
- Security testing tools are available and are trusted. Do not use free Internet-hosted tools.
- Review and control test procedures, scripts, and acceptance criteria.
- Test schedule that allows for resource availability.
- Configure system under evaluation as close as possible to its operational state without presenting a risk of disclosing sensitive data.
- Regression test paths with links to functions and data.
- Cross-reference of test descriptions to security requirements and any vulnerability found.

V&V Conduct
Execute security testing in a methodical and careful manner that will identify subtle anomalies that may lead to identification of an inherent vulnerability.
of the system. Order the test and verification activities to ensure that security vulnerabilities found later have not compromised previous testing. A good guide is to follow these steps:

- Get testing houses to perform required certification testing early and do necessary modifications before system validation testing.
- Conduct evaluation (inspection and analysis) and correct any inherent vulnerability found before performing extensive and costly security testing.
- When performing system validation testing or demonstration, complete testing, noting any anomalies found, to get the complete picture before reviewing results and system security impact.

**V&V Agility**

Verify and validate security to account for additional risks identified through security advisories from product suppliers or national computer emergency response teams organisations (for example, see United States Computer Emergency Readiness Team 2013). V&V must be able to adapt to this new scope and if possible extend testing to confirm the effectiveness of system protection.

**Reporting**

Besides just reporting on the results of testing of the system under evaluation, it is important to describe the impact that any ineffective controls have on the operation of the system and protected assets. Security verification and validation reports should include the following:

- Test environment and limitations
- Tests performed and impact of differences to real environment
- Test results and with details of severity of those that identify a deficiency
- Impact of deficiencies in terms of meeting the operational mission of the system

**Continued Revalidation**

Changes to the threat environment and ongoing introduction of new vulnerabilities necessitate periodic revalidation rather than non-regression of security controls. These vulnerabilities could come from sources like these:

- New system functional changes and enhancements
- Operating system patches
- Application software updates
- Vulnerability alerts
- Security incidents
- Regulatory changes

Consistent and regular patch management is a critical to keep evolving vulnerabilities under control. Beware that untested updates to security controls such as antivirus software and its pattern files, however, may generate untested false positives, which could cause the system to fail.

As summarised in figure 2, establish and maintain a set of security revalidation tests to ensure system upgrades and patches do not compromise either system functionality or security. Review and update these revalidation tests to address emerging threats and vulnerabilities and apply them periodically.

**Conclusions**

System-security verification and validation provides confidence in the effectiveness of security controls established to address known exploitable vulnerabilities. Apply a methodical approach that not only covers the explicit system-security requirement but also is adaptable to verify any emerging or discovered threats and vulnerabilities.

Exercise caution with the changing nature of vulnerabilities and maturing capability of threat agents. Security verification and validation is an ongoing activity even if there are no functional changes made to the system. Plan to revalidate the security of the system throughout its lifecycle. “Building it right” may not continue to align with “building the right thing” in the evolving world of security risks.

Figure 2. Security revalidation cycle

» continues on next page
An Approach to Integrate Security into a Systems Engineering Curriculum

Don Gelosh, donald.gelosh@incose.org

System security engineering has been defined as an element of systems engineering that applies scientific and engineering principles to identify security vulnerabilities and minimize or contain risks associated with these vulnerabilities. It uses mathematical, physical, and related scientific disciplines, and the principles and methods of engineering design and analysis to specify, predict, and evaluate the vulnerability of the system to security threats.

In addition, the Defense Acquisition Guidebook states that “in order to be cost-efficient and technically effective, System Security Engineering (SSE) must be integrated into Systems Engineering (SE) as a key sub-discipline” (US Department of Defense 2012, 13.7.6.2). To further emphasize this point, section 5.3.5 of the Program Protection Plan (PPP) Outline indicates that the Program Protection Plan should “Describe the linkage between system security engineering and the Systems Engineering Plan and answer the question, How will system security design considerations be addressed?” (US Department of Defense 2011, 18). These definition and policy statements suggest that system security engineering methods and techniques should be properly integrated throughout a system’s lifecycle—the design and build process from requirements to disposal—in order to produce and maintain a secure system. Therefore, in a similar fashion, system security engineering education should also be properly integrated throughout a course of study in systems engineering.

In their article entitled “Integrating Security into the Curriculum,” Cynthia Irvine, Shiu-Kai Chin, and Deborah Frincke (1998, 26) stated that “two important criteria for selecting outcomes for information security education are:

- The specific educational outcomes for security in a given educational program must be consistent with those of the larger engineering context.”

These criteria can be applied to system security engineering education in general. We must understand the current and future challenges to engineering secure systems and we must align the learning outcomes desired for security with the larger systems engineering context. Irvine, Chin, and Frincke (1998, 29) go on to suggest two approaches:

- “Computer security could be the focus of the curriculum, which would investigate the foundations and technical approaches to security in considerable depth.
- A computer science or computer engineering curriculum could choose to use computer security as an important property to be addressed in all coursework.”

Again, either or both of these two approaches could be used for integrating system security engineering into an existing systems engineering curriculum. We believe the latter approach is the most viable. Because we must build in security from the beginning and all throughout the system’s lifecycle, it seems only reasonable to build in and integrate security education throughout the entire program of study. With this overall idea of educational integration in mind, we will propose an approach to integrating system security engineering into a systems engineering curriculum.

Proposed Approach

Typically, a graduate or undergraduate systems engineering curriculum or course of study requires a set number of courses and the content in each required core course is a zero-sum game; if you add content, other content has to come out. We could simply develop a course on system security engineering and list it as an elective, but doing this does not guarantee that it will be taken
by anyone. In the case of system security engineering, we would like to include it without bumping other content because it affects many concepts in systems engineering. Therefore, we want to take a more inclusive approach and thread system security engineering concepts throughout the required courses. We can thread the concepts by examining what concepts make up system security engineering and then analyzing each concept and determining where and how it affects the systems engineering design and build process. The result is a mapping of where and how to integrate the concepts throughout the systems engineering curriculum.

The following is an example of how the proposed approach could be implemented. The system security engineering concepts here are notional and for illustrative purposes only. We expect different schools will have different concepts in mind based on how they define system security engineering and the particular expertise of their faculty and their research interests. Therefore, based on the definition provided above and generally accepted knowledge about system security engineering, a reasonable set of notional top-level concepts could include these:

- Security risk management
- Supply-chain risk management
- Cyber security
- Cryptography
- Information assurance
- Software assurance
- System assurance

The main activity in the proposed process is to analyze each of these concepts and determine where they would best fit into the systems engineering curriculum. We can analyze the concept by first defining the concept and then looking at its various lower-level concepts.

**System Security Engineering Concepts Analysis**

**Security Risk Management**

We can analyze Security Risk Management by looking at the definition of risk management in ISO/IEC 15288:2008 (29):

The purpose of the Risk Management Process is to identify, analyze, treat, and monitor the risks continuously. The Risk Management Process is a continuous process for systematically addressing risk throughout the life cycle of a system product or service. It can be applied to risks related to the acquisition, development, maintenance, or operation of a system.

It would be reasonable to extend the definition to include risks to the security objectives of the system. As with any risk assessment, we have to consider the probability of occurrence and the consequences. In the security domain, to calculate the probability of the risk, we have to consider threats to the system and the vulnerabilities of the system (both known and unknown). For the consequences, we have to consider the operational impact if the threat occurs. In some cases, the threats could evolve faster than the system can actually be constructed. The resulting risk analysis and mitigation plan may require the system to be designed with a certain amount of scalability and flexibility in order to respond to the new threats during its lifecycle. Because risk management is an integral part of systems engineering, most courses of study include the topic of risk somewhere in the curriculum. Using our proposed approach, risk management would be the best place to integrate security risk by covering the topics of existing threats, evolving threats, vulnerabilities, and operational impact assessments.

**Supply-Chain Risk Management**

The US Department of Defense defines supply-chain risk as “the risk that adversaries will insert malicious code into or otherwise subvert the design, manufacturing, production, distribution, installation, or maintenance of ICT [Information and Communications Technology] components that may be used in DoD systems to gain unauthorized access to data, to alter data, to disrupt operations, or to interrupt communications” (US Department of Defense 2010a, 13). The defense department defines supply-chain risk management as “the management of supply chain risk whether presented by the supplier, the supplied product and its subcomponents, or the supply chain (e.g., packaging, handling, storage, and transport)” (US Department of Defense 2010a, 12). Whereas this type of risk management is traditionally more concerned with risks associated with the availability and quality of components, in a system security engineering context it is concerned with things like counterfeit parts and malicious insertion of code into software, firmware, or logic-bearing hardware. Topics could include threats and vulnerabilities to supply chain stakeholders, vendor certifications, and counterfeit parts. These topics affect the entire acquisition lifecycle should be discussed whenever risk management is covered in the systems engineering curriculum.

**Cyber Security**

The US Department of Defense defines cyber security as “measures taken to protect a computer network, system or electronic information storage against unauthorized access or attempted access” (US Department of Defense 2010b, 10). Cyber security is very well known and there is a wealth of knowledge that can support an entire graduate curriculum. Perhaps the best we can hope to achieve in a systems engineering curriculum is a solid awareness of the concept and its
importance to systems throughout the lifecycle. Cyber security courses would be in addition to full graduate courses in cyber security topics that could be taken as electives.

Cryptography

In modern usage, cryptography goes beyond its historical role of securing communication systems. It has become a form of protection that endeavors to ensure any data sent by a sender is the same data received by the intended and authorized receiver and that the data cannot be accessed, changed, or denied transmission by an unauthorized third party. The modern concept of cryptography is very powerful because data protection can be applied to data that has to travel long distances or to data that may only have to travel very short distances within a system or within a component. Encrypted data protection is especially crucial if the systems are operating in hostile environments. Of course, the best way to provide encrypted data protection is to build it into the system from the beginning. Therefore, the recommended places to include cryptography are requirements, architecture and design courses, and perhaps courses on system integration.

Information Assurance

The concept of cryptography and the concept of information assurance are strongly related and interdependent. Information assurance can be defined as “measures that protect and defend information and information systems by ensuring their availability, integrity, authentication, confidentiality, and non-repudiation. This includes providing for restoration of information systems by incorporating protection, detection, and reaction capabilities” (US Department of Defense 2007). Some of the topics in a study of information assurance include security and survivability of networks and distributed systems; integrated security and dependability; and modeling and simulation. In some cases, information assurance can be achieved through nonmaterial solutions such as policy and training. However, in most cases, robust information assurance must be achieved through materiel solutions and built into the target systems. Therefore, information assurance should be considered across the system’s design-and-build process from requirements to operations, and information assurance knowledge should be addressed across a number of corresponding systems engineering courses.

Software Assurance

NASA (2008) defines software assurance as:

the planned and systematic set of activities that ensures that software life cycle processes and products conform to requirements, standards, and procedures. Software assurance includes the disciplines of: software quality (comprised of the functions of software quality engineering, software quality assurance and software quality control); software safety; software reliability; software verification and validation; and software independent verification and validation.

Some of the topics we typically include with software assurance are development and operating environments, code inspections, static testing, dynamic testing, penetration testing, element isolation, least privilege, multiple supplier redundancy, and release testing. In a manner similar to information assurance, software assurance techniques are implemented to ensure that the software in a system does what it is supposed to do and nothing more. Because most high-tech systems today are software intensive, software assurance should be implemented and integrated across a wide range of systems. Therefore, software assurance methods and techniques could be distributed across a wide range of systems engineering courses, with a concentrated focus in architecture and design and integration and test.

System Assurance

System assurance has been defined as “the justified confidence that the system functions as intended and is free of exploitable vulnerabilities, either intentionally or unintentionally designed or inserted as part of the system at any time during the life cycle” (NDIA System Assurance Committee 2008). Designing and building systems that function only as intended is certainly the main goal of all systems providers. The key here is building the system so it does not have other functionality present, intentional or otherwise, and remains free of exploitable vulnerabilities. The concept of system assurance must also be applied across the entire lifecycle of the system. “In systems engineering, assurance case development and maintenance should be executed as part of the stakeholder requirements definition, requirements analysis, architectural design, and risk management processes” (NDIA System Assurance Committee 2008). Therefore, system-assurance concepts, methods and techniques must be integrated throughout the curriculum, especially in courses that deal with requirements, architecture and design, risk management, integration and test, sustainability, scalability, and flexibility. Sustainability is very critical because we want to maintain fielded systems properly to ensure they stay free of exploitable vulnerabilities. Scalability and flexibility are also critical considerations because we want the system to have the capability to respond to evolving threats.
Conclusion

Due to the highly complex systems with emergent, undesired functionality that is difficult to predict or even fully test, achieving robust systems security requires a combination of artistic and technical skills. Designing and building secure systems is both an art and a science. However, some of the skills necessary to be a master security systems engineer can only be achieved through experience. Cynthia E. Irvine and J. R. Rao (2011, 19) emphasize the importance of experience:

Construction of highly trustworthy systems was and continues to be highly challenging and borders on an artisanal process: it requires leaders with considerable experience to head development teams, similar to the master masons who built cathedrals with a team of journeymen and apprentices during the Middle Ages.

There is a similarity between those who can really design and build secure systems, the master security systems engineers, and the master craftsmen of the Middle Ages. The master craftsmen knew the basics of their trade very well, but only when they added their artistic skills, achieved through years of experience, could they produce a masterpiece. Back then, becoming a master of any craft took several years of hard work. One would first start out as an apprentice to a master craftsman, learning the basics of the trade while responding dutifully to the beck and call of the master—one was basically an indentured servant. The apprentice lived with the master and his family, contributed to family chores, and had to work very hard to learn the trade. Living with the master day in and day out facilitated the efficient transfer of knowledge from the master to the apprentice. Then one day, the master decided the apprentice had learned enough and was ready to go out into world to learn more. The apprentice became a journeyman, or day laborer, who traveled from master to master, offering his services and learning different techniques from many master craftsmen. The journeyman began to develop his own artistic skills. Finally, the journeyman learned enough and saved enough money to open his own shop and become a master. How well the new master craftsman then performed depended on how well he learned the trade from both theory and experience through guided practice and how well he applied his artistic skills. Some wish we could develop security systems engineers in the same way, but in an accelerated manner.

Unfortunately the masters, those with in-depth engineering experience and expertise, eventually leave the workforce. When the masters leave, the apprentices and journeymen are left to fill the void, sometimes sooner than expected. We recognize attrition of the expert workforce as inevitable and therefore should endeavor to provide as much useful theory and practical experience to our systems engineering students as we can. Providing theory and practical experience especially holds true for the study and practice of system security engineering. Systems engineers should learn the theory behind the security concepts accompanied by practical experience in designing and building secure systems. When students graduate from a systems engineering program of study, they may understand the theory and have the tools and skills to learn more, but they are really apprentices. After a few years of work with a master security systems engineer, they are able to gain enough experience to evolve to journeyman status. Then they can start working on designing various systems, hopefully under the mentorship of a master. One day they can become masters with the necessary technical and artistic skills and experience. Academia must provide students with the best programs of study so their transition to master status can happen as quickly as possible. Having master-level systems engineers with a deep understanding of security is in the best interests of the academic community, the industrial and government communities, and especially the global systems engineering community.

References

CTI's CSEP exam preparation workshop, delivered over four days, is designed on leading edge adult learning principles, optimized for the type of content that needs to be mastered.

Throughout the course there is a strong focus on interaction, variety, the social aspects of learning and integration with the learner's existing knowledge framework.

“The instructor was very knowledgeable well-prepared, pleasant and accommodating. I would recommend this course to others.”

Delegate, Booz Allen Hamilton, USA

Consider scheduling this course at your premises for savings of up to 50% in comparison to attending a public delivery.

Certification Training International
PO Box 2385, Ringwood North
Victoria, 3134, Australia

Tel: +1 888 772 5174
Fax: +1 888 772 5191
www.certificationtraining-int.com
“The instructor was unquestionably a master of his material. Managed to pitch the material so that varying levels of expertise could get a great deal out of the course”
- delegate, Raytheon, USA

“Met my objectives very well. I achieved a better understanding of what should be done, rather than what is being currently performed”
- delegate, Carl Zeiss Optronics

“It was a very interesting course: I really learned a lot and it was a lot of fun ……”
- delegate, Germany

“The course added some very important fundamental knowledge to my understanding of requirements analysis and how to write a requirements specification”
- delegate, USA

### SYSTEMS ENGINEERING
for Technology-Based Projects and Product Developments

**Five day course**
Addresses systems engineering as it is understood and practised by leading acquirer, developer and supplier organizations worldwide. This training provides an integrated approach to the set of management and technical disciplines which combine to optimize system effectiveness, enhance project success and reduce risk.

**UPCOMING COURSE**
Las Vegas, NV
7 October - 11 October 2013

### REQUIREMENTS ANALYSIS & SPECIFICATION WRITING

**Five day course**
Requirements analysis and specification writing are sciences practised by many, mastered by surprisingly few. And yet, the payoff from achieving excellence in these areas is large. The two aspects, Requirements Analysis and Specification Writing, are treated as separate but related topics, each in a course of two and three days duration.

**UPCOMING COURSE**
Las Vegas, NV
26 August - 30 August 2013

Register now, or contact our friendly team with any queries.
No amount of money and magic will encourage a graceful transformation to success for a program entering verification work with a collection of badly prepared specifications.

1. A Problem

Even on a well-managed program executed by experienced professional engineers, requirements-verification work often bogs down in a failed attempt to deliver convincing evidence of compliance to the customer. This failure usually leads to an unnecessary loss of cost, schedule, and reputation. There are several things the contractor must do well to avoid this result, but there is one simple action in particular that is so seldom done as to merit special mention. Understanding this one thing could help enterprises overcome this problem on future programs so that requirements verification may be accomplished affordably. I happened upon this problem and a solution after some recent verification experiences with companies while also developing a third edition to my Elsevier Academic Press book System Verification.

2. The Needed Foundation

To frame the suggested action within the context of a foundation of necessary program actions, a brief discussion follows regarding an assumed enterprise organizational structure, an effective way of developing a set of good specifications, the need to establish a correlation between specifications and those responsible, the utility of establishing requirements linkage to supporting modeling efforts, and the capture of verification requirements.

2.1. Organizational Structure

If an enterprise can maintain a minimum necessary size, and if it can sustain a level of business where it has multiple healthy programs, then the enterprise should employ a matrix management structure. This structure combines a functional department structure (which builds and maintains programs) with a program structure that is composed of cross-functional teams. These teams are built around the product entity structure of the evolving system. These functional and program structures should include enterprise and program integration and optimization organizations in the form of an enterprise-integration functional department (under the systems-engineering or quality-assurance department) and a system-integration team for each program. The enterprise-integration functional department should be responsible for coordinating functional department structure evolution of a common system-development process and the related resources that will be deployed to programs. The system-integration team should be responsible for the conduct of systems engineering at the system level on each program. The objective is to treat the enterprise, programs, and product systems all as the systems they are.

2.2. Good Specifications

No amount of money and magic will encourage a graceful transformation to success for a program entering verification work with a collection of badly prepared specifications. The program work sequence of requirements, synthesis (design, procurement, and manufacturing), and verification—all of which must be accomplished within a context of good management—can only flow efficiently to a successful conclusion if that sequence is based on the sound foundation of a set of good specifications. This result will most often be realized through applying a comprehensive modeling approach to identify the entities and interfaces that must constitute the system and to identify the attributes that should be respected for those entities in requirements populating system, product entity, and interface entity specifications. Good engineering skill is then needed to identify appropriate values for those attributes appearing in requirements.

An enterprise should apply one modeling approach on all programs and that model should be comprehensive. This comprehensive model should be effective no matter how the design is to be implemented in terms of hardware, software, and people doing things. In a paper that was selected as best paper of 2009 in the INCOSE Systems Engineering journal, I identified three universal architecture description frameworks to cover problem-space modeling: (1) functional using some form of functional-flow diagramming, (2) the combination of modern structured analysis
(MSA) and process for architecture and requirements engineering (PSARE) referred to in the past as Hatley Pirbhai, and (3) the combination of unified modeling language (UML) and system modeling language (SysML). Since that time I have concluded that it is possible that a subset of unified process for the architecture frameworks of the United States Department of Defense and the United Kingdom Ministry of Defence (UPDM) would be useful in this way, if implemented using UML artifacts. In all cases the problem-space modeling should be supplemented with effective modeling methods for specialty engineering and environmental engineering, because many of these models as originally constructed did not include such methods. The concept motivating the use of a single comprehensive model is a desire to continue improvement through repetition and to encourage effective communication across the hardware–software boundary.

2.3 Responsibilities for System Definition and Specification

The program system team should be responsible for accomplishing the system-level modeling. This modeling will result in development of the top-level architecture for the system and publishing of the system and end-item specifications. The top-level cross-functional teams will form around those specifications. Ideally, the program would have pursued during early program proposal or planning work the preparation of a work breakdown structure (WBS) oriented around the evolving product structure, expanded into a statement of work. The statement of work provides top-level work statements appropriate to the WBS elements. The statement of work is further expanded into an integrated master plan and integrated master schedule for the program. Then as the system team establishes lower tier cross-functional teams, a team leader may be selected and presented with a specification and corresponding components of the plan and schedule as well as the staff needed to accomplish the related work and a budget.

Each cross-functional team must accept the responsibility to continue the modeling work initiated by the system team. In this work the cross-functional team employs the same comprehensive model to develop the next tier of architecture and specifications as used throughout the development effort. At each level the parent team becomes the integration-and-optimization agent for the lower team’s efforts.

2.4 Requirements Linkage to the Modeling Basis

The enterprise should evolve a way of assigning unique modeling IDs to every modeling artifact that may appear on modeling diagrams, such that requirements derived from these artifacts can be linked to those artifacts for traceability purposes. A program should not only establish traceability between modeling artifacts and the requirements derived from them but should also maintain a record in a configuration-controlled format. This can either be in the form of a system-definition report published from computer applications or in the content of the modeling applications employed. Those applications are configured to permit continuing work on modeling lower-tier elements while preventing change to previously approved higher-tier elements, unless a change has been approved in the same fashion as the original content was approved.

2.5 Verification Requirements

The author recommends that specification structures follow the six-section format defined in MIL-STD-961E. Section 3 of that standard, however, needs to be better coordinated with the modeling approach selected to respect the following kinds of requirements: (1) performance, (2) interface, (3) specialty engineering, and (4) environmental. For every product requirement in section 3 there should appear, of course, a verification requirement in section 4. The three kinds of specifications (system, item performance, and item detail) coordinate with the three kinds of verification work that must be accomplished on a program (system test and evaluation, item qualification, and item acceptance). The requirements appearing in section 3 apply to the features the design of the product must possess, but section 4 contains requirements for the verification process. The program must use the requirements in section 4 to determine the extent to which the product design complies with the content of section 3. In section 4 or 6 there should appear a requirements verification table. For every requirement in section 3, that table should list one or more requirements in section 4, along with the method that will be employed to determine compliance (test, analysis, examination, demonstration, or none) and the level at which the determination shall be accomplished (item, parent, or child).

The “none” method is included to clearly identify section 3 content that requires no verification activity, as in the case, for example, of a title-only paragraph. This same pattern can be applied to interface as well as product-entity specifications of both performance and detail type, of course.

3. Verification Planning, Implementation, and Reporting

3.1 Overview

Verification planning begins with the requirements-verification table that should appear in every specification. This table would ideally be derived from the database supporting all requirements and verification work. But this work can be accomplished with a pencil and paper in the extreme. The union of all of the requirements-traceability tables forms a program-verification compliance matrix, referred to by some as a requirements verification cross-reference matrix (VCRM). This matrix provides a basis for all verification planning, management, and reporting. The team structures can be assigned responsibility for the initial verification-planning work related to the specifications they were responsible for preparing.
Alternatively, the whole burden could be assigned to the program system team. In any case, someone from the system team should be assigned as the principal engineer for the whole process. In addition to the compliance matrix, a program is well served also by a task matrix to manage tasks and an item matrix to manage the items’ approach to audit readiness.

The first step in the planning of the verification process is to partition all strings in the verification-compliance matrix by method for a given item and then to determine a best way to collect the strings into some number of method-oriented tasks. Each task must have a principal engineer assigned, with the understanding that this person will prepare a task plan and procedure, acquire the necessary supporting resources for the task, accomplish the task in accordance with the plan and procedure as scheduled, and prepare a report of the results. The report for a system test-and-evaluation or item-qualification verification task must clearly state a conclusion about the degree of compliance of the product design relative to the requirements assigned to that task; or it must give a report about an item-acceptance verification task, indicating the degree of compliance of the product article manufacture relative to the requirements assigned to that task.

All of the content of this article so far has described what the author believes to be a fairly common description of development-program verification activity. Yet if this guidance is followed perfectly, the program can still experience a serious problem in completing the verification work in a timely and affordable fashion. So what is so different in what is being proposed?

### 3.2. The Difference

When a principal engineer is assigned to a verification task the engineer should be presented with available budget and schedule information, of course. But the engineer should also be given a clear list of the requirements that he or she is responsible for verifying. This list will be derived from the verification-compliance matrix that at this time will have had added the verification task number and principal's name. Further, the principal engineer should be informed that he or she must include a table in the task report that links each requirement to the precise content of the report where the evidence of compliance is presented (as in “paragraph 2.34,” “figure 3-4 zone 4F,” or “table 3-2”). Further, the principal should be informed that this information will be reviewed when the report is reviewed for approval. This review will ensure that anyone reading the report can easily reach a conclusion that the design is compliant with the requirement in each case. Ideally, the plan and procedure content would also clearly trace to the requirements, but that is of secondary importance so long as the report clearly communicates the results.

There are three patterns of customer behavior regarding review and approval of verification reports: (1) the customer reviews the reports as they are submitted, questions or approves them at that time, and no formal audits are performed; (2) the customer reviews and approves the reports as submitted, and at conclusion of the verification work holds an audit at item or system level; and (3) the customer does not express any interest in the details of verification while it is occurring, but holds a formal audit upon completion.

The common problem on many programs involving item qualification is that the reports do not clearly provide a description of the evidence of compliance linked to a specific requirement. The lack of a clear link between compliance evidence and requirements makes it difficult for a customer representative to reach a conclusion about the degree of compliance achieved. Often even system engineers working on program-verification status have difficulty completing verification-compliance matrix entries based on the content of the reports. What then happens is that an ever-increasing wave of uncertainty piles up, and this uncertainty supports the customer’s increasingly vocal demands for information about verification status.

When a program enters into this condition as it relates to the content of item-performance specifications, there is little that can be done but for the developer to request that the customer hold a functional configuration audit. In this audit, customer representatives will select some number of requirements and the contractor must provide the compliance evidence for those requirements. The contractor should select the people who prepared the reports, who will commonly be the engineers who did the related work, to explain the evidence to the customer representatives. A program, however, should never get into this condition by applying the guidance offered above. The critical point of focus for management is the review and approval of the reports and absolute adherence to the demand that every report include clear linkage between the evidence contained and the requirements it is related to.

If an enterprise insists on the policies discussed in this paper on its programs, then the engineers who prepare verification plans, procedures, and reports and do the verification tasks covered by these documents will learn how much easier the work is to accomplish when there is clear linkage between requirements and the evidence of compliance. It often is insufficient to simply write department procedures and assume that department personnel will perform as covered in those documents. The enterprise must first verify that its staff and programs comply with the enterprise’s requirements. Second—and equally as important—the enterprise must prove to the customer that its product complies with the requirements in the program specifications. This article offers one example of a way to include provisions in work practices that will ensure that the intended work will actually be carried out as defined.
In this column, I expand on my inaugural column to describe our different camps within INCOSE and the challenge within Technical Operations to relate these camps to effectively address the theme of this issue of INSIGHT, “The Buck Stops Here: Systems Engineering’s Responsibility for System Security.” I make no attempt to make our camps orthogonal to each other, but rather to stimulate discussion towards a shared vision that understands their roles, contributions, and value. With one exception, the camps I identify are well represented in our Technical Operations and local chapter working groups.

We have two camps within the practice of what many of us call classical systems engineering: (1) those members focusing on the technical processes across the lifecycle and (2) those focusing on what we call systems engineering management. The emphasis in both camps has been process focused. This is all captured quite explicitly in ISO/IEC 15288, Systems and Software Engineering—System Life Cycle Processes, and expounded upon in the INCOSE Systems Engineering Handbook, the latter serving as the basis of the systems engineering certification examination. A context for classical systems engineering is the industrial model for organizing repeatable work and has been the mantra for telecommunications, aerospace, and defense systems domains. Our standards initiative has its foundation in these two camps.

The early days of INCOSE were dominated by the aerospace and defense domains. In this context, classical systems engineering has been an empirical discipline, much as mechanical engineering was before the industrial revolution and electrical engineering was in the early 20th century before those two disciplines became based on science, once the science was understood and codified.

A breakout camp has emerged out of the proactive initiative of the Board of Directors to expand systems engineering through the Commercial Steering Board to nurture nontraditional domains that now include transportation, energy, and biomedical/healthcare. The nontraditional domain camp considers the classical systems engineering processes to be too fixed in the traditional domains and hence has to tailor it for application in the nontraditional domains.

Another breakout camp that has emerged from members initiative is model-based systems engineering (MBSE), to deal with the complexity of systems with formal methods and tools, moving systems engineering from an empirical to a model-based discipline. MBSE should result in fewer process defects compared to manual methods. Both traditional and nontraditional domains are represented in the set of challenge teams applying MBSE.

And then there is the camp of those members having a passion for soft systems, systems thinking, and systems science to deal with complex, stochastic, or nondeterministic systems for which it would be naïve or inappropriate to apply classical systems engineering. Personal observation based on attendance of annual symposia papers is that there appears to be more interest in soft systems and systems thinking among Sector II European members than among Sector I North American members. The attendance of papers at the 2012 International Symposium in Rome exceeded the capacity of the rooms, causing safety concerns by the Italian fire marshals. I would be interested to learn more about how these approaches are being received in Sector II African and Western Asian members and Sector III Asian and Oceania members.

A criticism of classical systems engineering is that it is more focused on processes than on successful outcomes. Significant research shows that sufficiently resourced systems engineering performed by competent systems engineers improve the odds for, but does not guarantee, successful
outcomes in the development of engineered systems. Likewise, MBSE is encountering resistance attributable to the inertia of the significant investment in legacy processes, methods, and tools. Right now the emphasis in the soft-systems, systems-thinking, and systems-science camp is focused on analysis; my personal vision is that the next phase move towards synthesis, the raison d’être of engineering.

A potentially missing camp is that focused on systems engineering leadership to provide the catalyst for the more successful outcomes. Systems engineering leadership leverages the mix of the skill sets of our different camps and perhaps credible domain knowledge to apply the right touch at the right time in the right way.

As I mentioned at the beginning of this column, the theme of this INSIGHT issue is “Systems Engineering’s Responsibility for System Security.” In the classical systems engineering approach using functional analysis, system security is regarded as an “ility,” much like reliability or safety, represented by its own engineering specialty domain having its own knowledge base, including our very own Systems Security Engineering Working Group. The tradition has been to treat system security as an add-on, or patch, to the basic system rather than considering it systematically up front. Established processes and methods relegate security to the set of nonfunction requirements (that is, constraints), rather than as part of the set of functional requirements with functions transforming inputs into outputs. The result is undesired inputs producing undesired outputs, or more appropriately, undesired consequences as well as undesired changes to the functions being performed by the system.

From the overall Technical Operations perspective, our challenge is to integrate the contributions of our different camps to successfully address the challenges of complex systems that are secure, safe, and resilient. We look forward to your engagement and contributions to move our discipline forward.

Members of the Transportation Working Group have been seeking case studies of the application of systems engineering in order to make a case to their colleagues for investing in systems engineering. To meet this need, the authors are assembling a library of case studies that illustrate the benefits of applying systems engineering ideas to transportation projects.

To ensure consistency, case studies are prepared following a standard process. Each case study is prepared either from authoritative documents in the public domain or by interviewing senior members of the project and checking the written case study with them. To ensure objectivity, each case study is written up by a team member not involved in the project.

The library currently includes 11 case studies, listed in the table below. One (no. 5) is a review of a research project and another (no. 11) illustrates the benefits of applying systems engineering at the corporate level. The other nine concern transportation-infrastructure-upgrade projects. One of these (no. 3) concerns air transportation, two (nos. 7 and 10) concern road transportation, while the remainder concern railways.

The majority of the findings of the case studies are “positive,” that is to say, they record the application of some aspect of systems engineering and conclude that benefits accrued. However a few contain “negative” findings in which some deficiency in the application of an aspect of systems engineering is recorded with a conclusion that the project may have suffered as a result. Some case studies contain findings of both sorts.

A single case study can only provide partial and circumstantial evidence for the benefits of applying systems engineering but a convincing case is emerging from the library as a whole. The simple analysis of the case studies provided in the table below illustrates this. The table lists the case studies and identifies whether they provide evidence for the benefits of key aspects of systems engineering (using processes defined in IEC 15228:2008). The following symbols are used:

- If the case study provides “positive” evidence
- If the case study provides “negative” evidence
- If the case study provides both sorts of evidence

Where the case studies do not report findings for a process, it cannot be inferred that the ideas underpinning this process were not applied, merely that they were not noted to be significant factors in the success or failure of the project.

A very strong case emerges from the case studies for the benefits to such major infrastructure projects of developing stakeholder and user needs and applying requirements analysis, with...
growing evidence for the benefits of verification, risk management, configuration management and information management.

<table>
<thead>
<tr>
<th>Case Study/Project</th>
<th>Process</th>
<th>Stakeholder Req.</th>
<th>Definition</th>
<th>Requirements Analysis</th>
<th>Architectural Design</th>
<th>Implementation</th>
<th>Integration</th>
<th>Verification</th>
<th>Transition</th>
<th>Validation</th>
<th>Project Planning &amp; Control</th>
<th>Measurement</th>
<th>Risk Management</th>
<th>Configuration Management</th>
<th>Information Management</th>
<th>Decision Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 West Coast Route Modernization (UK)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 SkyTrain control center upgrade and expansion (Vancouver, Canada)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 Prestwick Air Traffic Control Centre (UK)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 Docklands Light Railway Expansion (UK)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 NETLIPSE–European research project studying benefits of systems engineering applied to large infrastructure projects (EU)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Not related to a specific transportation project</td>
</tr>
<tr>
<td>6 East London Line Upgrade (UK)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 Santa Clara County Traffic Operations System and Signal Coordination (US)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 Jubilee Line Extension (UK)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 Jubilee Line Upgrade (UK)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 CityLink Replacement Control System (Australia)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 Network Rail Performance Modeling (UK)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Not related to a specific transportation project</td>
</tr>
</tbody>
</table>

The library is available on the Transportation Working Group’s Connect site. Further case studies are always sought, with some in progress. To suggest a case study, or for further information, please contact one of the authors. 🌐

---

Elliott et al. continued

Report on the 2013 Workshop of INCOSE’s Systems Engineering and Architecting Doctoral Student Network

Ricardo Valerdi, ricardo.valerdi@incose.org; Donna H. Rhodes, donna.rhodes@incose.org; and Cecilia Haskins, cecilia.haskins@incose.org

The purpose of the INCOSE-sponsored Systems Engineering and Architecting Doctoral Student Network (SEANET) is to foster excellence in research in systems engineering and related disciplines. This year’s workshop was held in conjunction with the 11th Conference on Systems Engineering Research (CSER) in Atlanta, Georgia (US). The workshop was attended by 46 individuals, of which 34 were participating as doctoral students representing the largest SEANET group since its inception in 2005.

Faculty and students from eight countries were represented including the United States (Georgia Tech, Johns Hopkins, MIT, Missouri University of Science and Technology, Purdue, Stevens Institute, Texas A&M, University of Alabama Huntsville, University of Arizona, University of Southern California, Penn State and University of Texas El Paso), Germany (Technical University of Darmstadt and RWTH Aachen University), UK (University of Bristol), China (Beihang University), Brazil (Engelflux), Sweden (Chalmers University of Technology), Norway (Norwegian University of Science and Technology), Italy (University of Modena and Reggio Emilia), the Netherlands (University of Twente), and France (Ecole Centrale Paris). Additionally, researchers from industry included Siemens Corporation, Thales, and ENGEFLUX.

Overview of the 2013 Workshop

Building on the success of previous workshops (Rhodes and Valerdi 2007; Valerdi, Rhodes, and Kalawsky 2009; Valerdi and Rhodes 2010, Valerdi, Rhodes, and Dagli, 2012), the day’s agenda provided a balance between
keynote presentations, short vignettes on research, and opportunities for networking and knowledge sharing.

Donna Rhodes, principal research scientist and senior lecturer at the Massachusetts Institute of Technology, and director of SEANET, presented the history and motivations for the doctoral network. She shared results of student surveys from prior SEANET workshops and discussed some of the trends and positive indicators for doctoral research in the field. SEANET was initiated in 2005 to foster excellence in systems engineering research and encourage knowledge sharing across the research community (Rhodes 2005).

Barry Boehm, professor of software engineering at the University of Southern California, shared his perspective on current trends in research such as data mining and big data, smart sensors and smart cities, the Internet of things, cloud computing, service-oriented architectures, cybersecurity, multicore chips, and recommender systems. He also discussed two wild-card trends that could drive future opportunities for research: autonomous software and combinations of biology and computing. He also emphasized the ability to “learn how to learn” during your doctoral program since it will help you to keep up with future trends.

Cecilia Haskins, associate professor at the Norwegian University of Science and Technology, provided an overview of doctoral research in Europe. The European countries with the most academic programs (bachelor's, master’s, and PhD) in systems engineering are the UK, France, and Germany. Many of these are coupled with industrial-engineering or industrial-design departments. There are also some Europe-based journals covering systems engineering topics such as the *Journal of Systems and Software*, *Reliability Engineering & System Safety*, and the *European Journal of Industrial Engineering*, to name a few.

Paul Martin Gibbons, University of Bristol and Gatwick Airport, shared his research journey towards the EngD which began in 1996. Paul participated in the 2007 SEANET workshop and completed his EngD in 2011. He shared twelve typical viva (that is, doctoral defense) questions that all students should be prepared to answer:

1. Why did you choose this topic for your doctoral study?
2. How did you arrive at your conceptual framework?
3. How did you arrive at your research design?
4. How would you justify your choice of methodology?
5. Why did you choose XYZ as your main instrument(s)?
6. How did you select your respondents/materials/area?
7. How did you arrive at your conceptual conclusions?
8. How generalizable are your findings and why?
9. What is your contribution to knowledge?
10. How would you to critique your thesis?
11. What are you going to do after you gain your doctorate?
12. Is there anything else that you would like to tell us about your thesis that does not appear in the written manuscript?

Adam Ross, research scientist at MIT, shared ideas on his journey of knowledge creation. Three questions are important in determining your criteria for success as a doctoral student:

1. Why are you in a PhD program?
2. Are you worthy of a PhD?
3. What must you do to complete your PhD?

Dr. Ross also noted that the PhD process is both epic and humbling, requiring intense efforts and mastery over forces in tension. Some tensions include creativity vs. productivity, synthesis vs. analysis, breadth vs. depth, the present vs. posterity. Some ways to address these tensions are to seek advice, take initiative, pursue balance, give yourself space to think and reflect, find the right support and tools, identify the right toys or games that give you an opportunity to be creative and even fail, and don’t forget to be a meta-thinker. Getting a PhD is like passing through a gateway with a hand full of keys: some doors open and some doors close.
The workshop also included four breakout groups on topics of interest to doctoral students. Each group had the benefit of an interactive question-and-answer forum with a faculty member who facilitated 40-minute discussion sessions.

Adam Ross led the discussion on “Scoping, Discovery, and Literature Search” and had the following observations:

- There is a lot of variation in how people acquire dissertation topics, ranging from having a topic handed to you by your advisor, to students developing their own ideas and choosing a compatible advisor.
- Dissertation topic selection is driven by timing constraints (a three-year hard stop for some), funding, risk of advisor leaving, and emergent requirements from sponsors.
- The sheer size of the literature is overwhelming. The more you read the bigger the literature seems. The 80% solution is to keep reading until you reach a saturation point among commonly appearing authors in a certain specialty.
- Lack of crystallization of research questions goes hand in hand with proper scope development and determines the focus of the literature review.

Paul Gibbons led the discussion on “Designing and Applying Research Approach and Methods” and had the following observations:

- The choice of research method is an anchoring type of activity that helps determine the basis for the validity of the research.
- Research method also drives the decision to collect certain types of data.
- Mixed methods can contribute in a step-wise way as the research matures.
- Since a PhD is a doctor of philosophy, it is important to remember that part of the philosophy is about how to do research.

Jo Ann Lane of USC led a breakout group on “Performing Data Collection, Interviews, Modeling, and Experiments” and had the following observations:

- Many of the students are involved in research teams where plenty of data are available.
- Access to data can make or break one’s ability to validate their research question.
- It is important also to look for data surrogates like old or retired projects that have publicly available data.
- If you are using sensitive or classified data you need to ask yourself whether the results will be publishable.
- When collaborating with companies you must be able to answer two questions: What is in it for them? How much effort will it take the company to support you?

Carlee Bishop of Georgia Tech led a breakout group on “Research End Game, Validation, Publishing, and Finding Opportunities” and had the following observations:

- Conference papers could end up as chapters in one’s dissertation.
- Students need to involve their dissertation committee well before the defense.
- Students need to be conscious of any publication requirement from their department.
- It is important to find champions for your research since you will have a captive audience and transition partners.
- Serve as a reviewer for journals: it is good practice.
- One way to find an appropriate journal for your work is to look at where your advisor has published in the past.
- Be conscious of copyright restrictions on conference and journal publications.
- Impact factor may be an important criterion for selecting a journal.
- Completed PhD dissertations in your department supervised by your own advisor are good indicators of successful examples.
- An obvious place to start with a publication is to write a literature-review paper and submit it to a conference.

These interactive sessions provided the SEANET participants the opportunity to openly share their thoughts and concerns about specific aspects of their PhD journey. Many of them also participated in the poster session that evening which provided them with the opportunity to showcase their work to CSER attendees during the welcoming reception. Due to the continued success and increasing participation, we plan to organize have the next SEANET workshop in conjunction with CSER 2014 at the University of Southern California in Los Angeles, California (US).

References
Are you ready to advance your career in systems engineering? Then look into INCOSE certification and set yourself apart. We offer three levels of certification for professionals who are ready to take charge of their career success.

Apply for INCOSE Certification Today!

Visit www.incose.org or call 800.366.1164
INCOSE Spotlight

INCOSE Spotlight on... Wolter J. Fabrycky

Sandy Young, info@incose.org

Name: Wolter J. Fabrycky, PhD, PE
Titles and organizations: Lawrence Professor Emeritus at Virginia Tech and Chairman of Academic Applications International
Place of birth: Queens, New York (US)
Current residence: Blacksburg, Virginia (US)
Domain: Industrial and Systems Engineering
Studied: Industrial and General Engineering

Year joined INCOSE: 1990 (Charter Member No. 21)
Roles in INCOSE: Board of Directors 1995–97, Associate Editor for the INCOSE Systems Engineering journal since 1998, Fellow Award in 1999, Pioneer Award in 2000, and the Accreditation Board for Engineering and Technology (ABET) Board of Directors since 2008
Years in systems engineering: 45

What did you want to do for a job when you were a little kid? I wanted to understand how things worked, but I also wanted to live and work on a farm. My dad became a sharecropper in Florida in the 1930s, but that soon failed. Then in 1949, he moved us from New York to a worn-out farm in Arkansas, where I finished high school by taking vocational agriculture.

What inspired you to become a systems engineer? I was appointed in 1968 by Virginia Tech [VT] Dean of Engineering Willis Worcester to organize an interdepartmental faculty group for systems engineering. At the time, I was directing the VT Operations Research Center. It impressed me that our dean would organize engineering into a matrix with 13 classical departments overlaid by five interdisciplinary Technical Interest Groups to enhance collaborative interaction.

This led me to learn about and develop systems engineering, and then to lead this emerging interdiscipline for VT and beyond. I was inspired by the challenge to extend the largely analysis- and optimization-based specialty of operations research to consider the entire system lifecycle. Our emphasis was on operational problem avoidance through the early employment of design synthesis over the system lifecycle. We demonstrated that systems engineering is more than systems analysis. Unfortunately, too often the terms “systems analysis” and “operations research” are incorrectly referred to as systems engineering.

What was your path to becoming a systems engineer? Becoming an Arkie farm boy was pure drudgery. So I enrolled in pre-engineering in 1951 and transferred to Wichita State to finish a bachelor’s degree in industrial engineering in 1957. At the same time, I worked for Cessna Aircraft. I completed a master’s in industrial engineering at Arkansas in 1958, and stayed on to teach engineering graphics and industrial engineering until 1960. Committed to academia, I earned a PhD in general engineering at Oklahoma State in 1962 and stayed on as an OSU Okie faculty member. In 1965, I became a Hokie professor at Virginia Tech, which quickly led to my emphasis on systems engineering.

Can you briefly explain your role as founding chairman of systems engineering at Virginia Tech? My role as systems engineering chairman was to define systems engineering as a graduate-level academic interdiscipline, organize an interdepartmental advisory committee, and select practicing engineers as students. In 1976, I passed the chairmanship on to Benjamin Blanchard, so I could join the VT central administration as university dean of research. That position included responsibility for the Virginia Agricultural Experiment Station. It was interesting how my farm and VoAg [vocational agriculture] background suddenly became very important!

What may others not know about the book Systems Engineering and Analysis that you coauthored? I want to let all systems engineers know that Systems Engineering and Analysis, first published in 1981, is now in its fifth (30th anniversary) edition. Many systems engineers are still using and making references to the out-of-date editions. Also, note that my venerable colleague Blanchard and I are beginning to seek a coauthor for the sixth edition, as encouraged by Pearson Prentice Hall.
Spotlight on… Wolter J. Fabrycky  continued

What professional accomplishments are you most proud of? I’m proud to be with Pearson Prentice Hall as a textbook author and series editor for more than 50 years (see INSIGHT vol. 15, no. 1). Also, I was happy to discover the Design Dependent Parameter (DDP) paradigm for “stumbling through” the system design space, permitting mutually exclusive design alternatives to be evaluated on an equivalent basis over the system lifecycle. Parameters are rarely mentioned, and DDPs are completely ignored in operations research, but are now becoming the key to integrating the full potential of operations research within the systems engineering process.

What trends do you see in your domain? There is a rapidly emerging interest within the engineering profession to integrate system concepts and thinking into most domains of engineering. I call this Domain Centric Systems Engineering as contrasted to Systems Centric Systems Engineering, INCOSE style. That trend is timely and good to observe. But, I see and fear the awarding of graduate degrees in systems engineering to individuals who have not earned an ABET-accredited degree in engineering, or its equivalent. Exceptions are to be expected, but they should not become commonplace.

How has INCOSE benefitted you? INCOSE began as NCOSE 18 years after VT awarded its first systems engineering master’s degree. Since then, INCOSE has secured global validation for the interdiscipline of systems engineering, and confirmed the wisdom of Dean Worcester at VT, as well as of Dean Martin and Wayne Wymore in Arizona, who first established systems engineering in academia. Further, the advent of INCOSE gave me the opening to establish an international honor society for systems engineering, the Omega Alpha Association (see http://www.omegalpha.org).

What do you like to do outside of work? I’m one of those individuals who has little interest in things outside of work and my family. One exception is that I admire and follow the Austrian School of Economics, through the Ludwig von Mises Institute and the Foundation for Economic Education. This school is, in my opinion, the best economic underpinning for systems engineering and also for the subject of engineering economics, for which I coauthored the Pearson Prentice Hall Engineering Economy textbook, now in its ninth edition.

I also purchased a 95-acre farm here in Hokieland in 2000. I have now adopted a mission statement involving the production of natural produce under the business name iFarmVA. Perhaps, this is as an emotional response to my time on farms in Florida and Arkansas.

Valerie Gundrum Engineering Scholarship Fund

The Broome Community College Scholarship Foundation in Binghamton, New York (US), has established the Valerie Gundrum Engineering Scholarship Fund. The scholarship honors the life of Valerie and will be awarded to a female entering engineering who possesses the qualities of leadership, is focused in her studies, is an inspiration and mentor to others in regards to her studies, and who has a genuine love and dedication when helping others. The INCOSE Foundation will receive donations from individuals to support this scholarship fund. The Foundation Board voted to honor Valerie with USD 2,000 donation to the scholarship fund this year and to make continuing donations to the scholarship. Donations may be made to the fund by designating “Valerie Gundrum Fund” at http://www.incose.org/about/foundation/.

Nominations Now Open for the David Wright INCOSE Leadership Award

INCOSE and the INCOSE Foundation are accepting nominations for the inaugural David Wright INCOSE Leadership Award. Nominations are due by 1 November. In consideration of the values held by David Wright and to honor his vision for INCOSE to develop leaders, this two-year award will:

• Allow an INCOSE member to participate in an established INCOSE collaborative project;
• establish a mentor relationship with the INCOSE President-elect; and
• provide funding for attendance at the collaborative partner organization’s conference, INCOSE’s International Workshop and International Symposium (in the years of the award).

• For full details, please visit www.incose.org/about/foundation or contact Holly Witte at holly.witte@incose.org.
This book brings to life the 1960s and 70s, when social, political, and technological change was rife in the United States. Engineers, the guys so commonly thought of as wearing white short-sleeve dress shirts, ties, and pocket protectors, with little interest in the world around them, were becoming part of the discussion. Were engineers responsible for the impact of the technology they brought forward? Should they be involved with educating the public and the discussions underway on how technology was changing lives and careers? Or were they just providers and it was up to the users to address the implications? Were defense spending and NASA missions the best way to use the human and financial capital? Or should funding be pointed towards programs that directly improved lives?

Engineering grew after World War II as technology applications increased and defense spending stayed high as the new concerns of what would become the Cold War mounted. Matthew Wisnioski explores how these times impacted professional organizations, corporations, government, and academia—how engineers viewed themselves and how they responded to the public’s view. While many engineers saw themselves as simply working on technology and were little concerned with the impact of that technology, others were asking questions about the way engineers were used and trained, and about how research was performed at universities. Should the focus move from defense to other fields? Demonstrations at IEEE meetings by members concerned about corporate and government influence focused new light on how some engineers viewed themselves and the profession. While some engineers saw this as a time with unlimited opportunity with the peak of the Cold War defense spending, increases in the NASA budget for the race to the moon, spending for equipment in Vietnam, and an overall booming economy. For others this was a time of despair, feeling that civilization was being brought to the edge of destruction, with concerns about the impact of Vietnam on the US and its citizens. The tension between the profession, those that employed engineers and those that taught them, is not only interesting, but relevant in today’s society, where professional societies still need to balance the needs of their members with the needs of those that employ them and universities that train them. Today we still discuss the focus on government-funded research and the implications for those doing the work.

The tension on university campuses to determine the best way to educate engineers to be good citizens leads to challenges from all of the participants, students, professors and administrators. Throw in government agencies and you’ve got plenty of room for conflict. How well rounded does an engineer need to be? Is there time in the program to address the social along with the technical?

Wisnioski also describes an interesting partnership that brought engineers and artists together. Experiments in Art and Engineering was an organization founded in 1969 bringing together artists and engineers to create and shape technology.

At that time words like innovation were beginning to become part of corporate and education discussions. With the same problem we face today, how does one teach innovation? Can it be...
Carbon Footprint Analysis: Concepts, Methods, Implementation, and Case Studies

By Matthew John Franchetti and Defne Apul
(ISBN 978-1-4398-5783-0)

Reviewed by Lawrence D. Pohlmann, pohlmann@incose.org

“While the topic is timely and urgent and somewhat haphazardly practiced, the relevant content and skill set for carbon footprint management has been published in bits and pieces[.]

Target Audience. This book is primarily technical practitioners directly involved with carbon-footprint assessment and management. The book aspires to be a handbook or reference book for processes and techniques. The authors also consider the book suitable for academic or instructional use. A companion set of “active learning exercises” is under development. One back-cover reviewer states that this book is “the first comprehensive engineering and technically based treatment of all the associated issues in one book.”

The Authors. Dr. Matthew Franchetti is an assistant professor of mechanical, industrial, and manufacturing engineering at the University of Toledo (Ohio, US). He is also the director of the university’s Environmentally Conscious Design and Manufacturing Laboratory. He works with a research group that has assisted more than 100 companies with various aspects of sustainability assessments. Dr. Defne Apul is an assistant professor of civil engineering at the University of Toledo. She focuses on sustainability engineering research and teaching, including sustainability lifecycle engineering. From this reviewer’s viewpoint (being largely naïve on carbon-footprint analysis and sustainability), the authors seem qualified to write on the topic. Their advocacy of a systems approach indicates knowledge of systems engineering basics.

Book Structure and Content. The structure and content are foreshadowed by the book’s subtitle. Concepts are addressed in section I, “Why Carbon Footprint Analysis and Reduction?” This section (about 10% of the book) provides background and introduces basic concepts and terminology, and summarizes applicable regulations. Readers from outside this specialization may find this section useful for understanding these basics.

Methods, addressed in section II, focuses on various accounting methods for assessing and tracking the emission of greenhouse gases. The key message is that there are multiple ways to think about footprint assessment and management.

Implementation, addressed in section III, is the meat of the book (about 45%). The section strongly advocates a systems approach to total lifecycle assessment. A wide range of prescriptive guidance is provided (such as different perspectives from which to view the situation, possible organizational initiatives). This section will be repeatedly useful for assessment practitioners.

Case Studies, in section IV (about 20% of the book), includes nicely diverse examples from higher education (a university), manufacturing (a compressor manufacturer), health care (a hospital), and construction (a construction support company). Assessment approaches, assessment data, and carbon-footprint-reduction strategies and recommendations are made for each case. Readers may find this section useful for appreciating some of the potential problem and solution dimensions for their specific situation—and for other types of situations.

Usability. I found the book to grow on me as I developed this review. Is this book usable as a handbook or reference book? Yes, it has lots of useful guidance for practitioners. Is it...
Should You Buy This Book? This book will be most useful for:
- Technical practitioners directly involved with sustainability assessments;
- Those establishing best practices for small-footprint, sustainable systems; and
- Systems engineers who define and document sustainability requirements.

In Closing. I believe it is inevitable that the systems engineering community will (must?) become increasingly involved with developing more sustainable, “greener” systems. Books like this will help technical practitioners—including systems engineers. The book strongly advocates the type of big-picture systems approach that is the forte of our discipline. While this book is not a primer for beginners, managers, and executives, books that satisfy this need may be more broadly useful to the INCOSE community—and may exist. Maybe it is time for an INCOSE Sustainability Working Group to be established and assigned to assemble a set of information that communicates sustainability basics.

“Carbon Footprint Analysis continued
a good handbook or reference book? Not yet! I would rate it a B minus. It needs more figures and fewer “micro-font” figures. A list of figures and tables is needed. A better list of terminology and abbreviations should be easier to find. Key messages (in the main text, figures, and tables) need to stand out more. Guidance needs to be made easier to locate, by means of more headers and highlighting. These shortcomings could be easily fixed in a second edition.

Orchestrating Human-Centered Design
By Guy André Boy
Reviewed by Lawrence D. Pohlmann, pohlmann@incose.org

“Human Centered Design involves a broader perspective of what we want to do with technology, organizations and people [. . .] to cope with the current finance driven organizational systems [. . .].”

Target Audiences. While many authors say things like “this book is intended for . . .” Dr. Boy does not. From the title, you might expect this to be a how-to book for mid- to senior-level systems engineers, human factors engineers, and process engineers. This is reasonable. I believe he is talking to people mature enough to think about their roles philosophically, rather than merely procedurally. For academics who deal with humans in systems concerns, the book is rich in references and theory of design processes. Lastly, I believe this book is also targeted at corporate chief technology officers, program chief engineers, and systems engineering managers who must make decisions on product-design philosophies, research-and-development investment strategies, and human-resource allocations on product-line and individual program developments. Included in all of these are decisions on when, how much, and how we think about the human elements (builders, transporters, installers, operators, maintainers, and others) who will be involved in some way with our systems over the system lifecycle.

The Author. Dr. Guy André Boy is a university professor at Florida Institute of Technology and director of that institution’s Human Centered Design Institute. He is also Chief Scientist for Human Centered Design at the NASA Kennedy Space Center. He was educated in France; he has over a 30-year career in the interaction of humans and machines and the integration of humans and machines. He is a prolific author, and has an extensive set of publications in both French and English, including 10 books. He has supported and consulted with both industry and government. The extent of his experience allows him to bring engineering, managerial, and philosophical perspectives to the discussion. Additional information on Dr. Boy is available through his personal website at my.fit.edu/~gboy. He is obviously very well qualified as an author in this engineering and management area.

Content and Structure. A continuing theme throughout the book is lamenting that we are too entrenched in finance-driven, short-term thinking—and that this is not in the best interest of our system designs, our longer-term company successes, our national interests, and our longer-term interests for society at large. A second theme is that in building our
Orchestrating Human-Centered Design continued

In systems, especially life-critical systems, we need a balanced consideration of technology, organizations, and people (he labels this the TOP model)—and that this balance can be “orchestrated.” Thirdly, successful human-centered design (HCD) is best achieved through thinking across the total system lifecycle from six perspectives: life critical systems + cognitive science + complexity analysis + advanced interaction media + modeling and simulation + organization design and management. Each chapter discusses one of these in detail and uses a wide range of supporting stories and anecdotes.

Usability. This book provides a rare opportunity to take in and benefit from the more than 30 years of experience of a man who has achieved elder-statesman status in thinking about how to appropriately and efficiently include human considerations across the lifecycle of systems. The numerous anecdotes and stories provide vicarious experience to those engineers, managers, and executives who are experienced enough and willing to benefit. The book is organized around the six human-centered design perspectives listed above. The nearly stream-of-consciousness writing style is challenging. Some readers may have difficulty with this approach.

Should You Buy This Book? If you are part of one of the target audiences suggested above, consider the book. For INCOSE members, the book should be required reading for the Human Systems Working Group. It will also be of interest to the Complex Systems Working Group, the Systems Science Working Group, and the System of Systems Working Group. I would also encourage Corporate Advisory Board members of large companies that have matrix management to recommend the book to their organizations that are responsible for human factors and crew systems.

In Closing. This is a book for thinkers. It is not a process recipe book for junior practitioners. It is far ranging. It confronts serious and difficult issues on how we design systems, especially life-critical systems. It suggests ways to think and act differently. It is not read-a-chapter-at-lunch or -bedtime book. It requires a serious approach to reading and use. I believe the book is best used in a repeated read-a-little, think-a-little manner over an extended period by senior decision-makers who are dealing with serious and far-reaching issues for designing and developing complex, human-in-the-loop systems. I like the book! I look forward to spending more time with it over the next few months!

“HCD is not about human factors and ergonomics that is used when systems are already designed and developed . . . HCD is about reinventing engineering and design in to a single discipline that integrates technology, organization, and people.”
The Systems Engineering journal is intended to be a primary source of multidisciplinary information for the systems engineering and management of products and services, and processes of all types. Systems engineering activities involve the technologies and system management approaches needed for

- definition of systems, including identification of user requirements and technological specifications;
- development of systems, including conceptual architectures, tradeoff of design concepts, configuration management during system development, integration of new systems with legacy systems, integrated product and process development; and
- deployment of systems, including operational test and evaluation, maintenance over an extended lifecycle, and re-engineering.

Systems Engineering is the archival journal of, and exists to serve the following objectives of, the International Council on Systems Engineering (INCOSE):

- To provide a focal point for dissemination of systems engineering knowledge
- To promote collaboration in systems engineering education and research
- To encourage and assure establishment of professional standards for integrity in the practice of systems engineering
- To improve the professional status of all those engaged in the practice of systems engineering
- To encourage governmental and industrial support for research and educational programs that will improve the systems engineering process and its practice

The journal supports these goals by providing a continuing, respected publication of peer-reviewed results from research and development in the area of systems engineering. Systems engineering is defined broadly in this context as an interdisciplinary approach and means to enable the realization of successful systems that are of high quality, cost-effective, and trustworthy in meeting customer requirements.

The Systems Engineering journal is dedicated to all aspects of the engineering of systems: technical, management, economic, and social. It focuses on the life-cycle processes needed to create trustworthy and high-quality systems. It will also emphasize the systems management efforts needed to define, develop, and deploy trustworthy and high quality processes for the production of systems. Within this, Systems Engineering is especially concerned with evaluation of the efficiency and effectiveness of systems management, technical direction, and integration of systems. Systems Engineering is also very concerned with the engineering of systems that support sustainable development. Modern systems, including both products and services, are often very knowledge-intensive, and are found in both the public and private sectors. The journal emphasizes strategic and program management of these, and the information and knowledge base for knowledge principles, knowledge practices, and knowledge perspectives for the engineering of systems. Definitive case studies involving systems engineering practice are especially welcome.

The journal is a primary source of information for the systems engineering of products and services that are generally large in scale, scope, and complexity. Systems Engineering will be especially concerned with process- or product-line–related efforts needed to produce products that are trustworthy and of high quality, and that are cost effective in meeting user needs. A major component of this is system cost and operational effectiveness determination, and the development of processes that ensure that products are cost effective. This requires the integration of a number of engineering disciplines necessary for the definition, development, and deployment of complex systems. It also requires attention to the lifecycle process used to produce systems, and the integration of systems, including legacy systems, at various architectural levels. In addition, appropriate systems management of information and knowledge across technologies, organizations, and environments is also needed to insure a sustainable world.

The journal will accept and review submissions in English from any author, in any global locality, whether or not the author is an INCOSE member. A body of international peers will review all submissions, and the reviewers will suggest potential revisions to the author, with the intent to achieve published papers that

- relate to the field of systems engineering;
- represent new, previously unpublished work;
- advance the state of knowledge of the field; and
- conform to a high standard of scholarly presentation.

Editorial selection of works for publication will be made based on content, without regard to the stature of the authors. Selections will include a wide variety of international works, recognizing and supporting the essential breadth and universality of the field. Final selection of papers for publication, and the form of publication, shall rest with the editor.

Submission of quality papers for review is strongly encouraged. The review process is estimated to take three months, occasionally longer for hard-copy manuscript.

Systems Engineering operates an online submission and peer review system that allows authors to submit articles online and track their progress, throughout the peer-review process, via a web interface. All papers submitted to Systems Engineering, including revisions or resubmissions of prior manuscripts, must be made through the online system. Contributions sent through regular mail on paper or emails with attachments will not be reviewed or acknowledged.

All manuscripts must be submitted online to Systems Engineering at ScholarOne Manuscripts, located at http://mc.manuscriptcentral.com/SYS. Full instructions and support are available on the site, and a user ID and password can be obtained on the first visit.
Recently INCOSE renewed its relationship with John Wiley & Sons, Inc., which describes itself on its website as follows:

Wiley is a global provider of content-enabled solutions that improve outcomes in research, education, and professional practice. Our core businesses produce scientific, technical, medical, and scholarly journals, reference works, books, database services, and advertising; professional books, subscription products, certification and training services and online applications; and education content and services including integrated online teaching and learning resources for undergraduate and graduate students and lifelong learners.

Founded in 1807, John Wiley & Sons, Inc. (NYSE: JWA, JWb), has been a valued source of information and understanding for more than 200 years, helping people around the world meet their needs and fulfill their aspirations. Wiley and its acquired companies have published the works of more than 450 Nobel laureates in all categories: Literature, Economics, Physiology or Medicine, Physics, Chemistry, and Peace. Wiley’s global headquarters are located in Hoboken, New Jersey, with operations in the U.S., Europe, Asia, Canada, and Australia. The Company’s website can be accessed at http://www.wiley.com.

Wiley now serves INSIGHT in two roles: as advertising editor and distributor. Wiley has assigned Roland Espinosa to be the advertising representative for INSIGHT. He can be contacted by e-mail at respinosa@wiley.org and by telephone at +1-201-748-6819. In its role as distributor, Wiley is responsible for printing and distribution of the hard-copy issues of INSIGHT worldwide. We welcome the expanded partnership with Wiley in supporting INCOSE and its members with these important services.

INCOSE has recruited Scott Flander, who is a reporter for the Philadelphia Daily News and has authored two crime novels about Philly street cops, Four to Midnight and Sons of the City, to provide coverage of the keynote speakers at the international symposium in Philadelphia. We are fairly certain that the stories you will read in INSIGHT will be exciting and true and that the names will not have been changed to protect the innocent, or maybe not!

We now have a full slate of issues of INSIGHT planned through July 2015 with theme editors who come from France, the United States, Australia, and the Netherlands. We still have an opening for someone to provide coverage of our keynote speakers at the 24th Annual INCOSE International Symposium in Korea.

**Upcoming submission deadlines and themes for INSIGHT**

<table>
<thead>
<tr>
<th>Issue</th>
<th>Submission Date for General Articles</th>
<th>Theme</th>
<th>Theme Editors</th>
</tr>
</thead>
<tbody>
<tr>
<td>September 2013</td>
<td>8 July 2013</td>
<td>2013 International Symposium Coverage: Philadelphia, PA (US)</td>
<td>Scott Flander</td>
</tr>
<tr>
<td>April 2014</td>
<td>15 February 2014</td>
<td>Standards</td>
<td>Ken Zemrowski</td>
</tr>
<tr>
<td>December 2014</td>
<td>15 October 2014</td>
<td>Model-Based Conceptual Design: Engineering the Problem Space</td>
<td>Kevin Robinson, Quoc Do, and Michael Waite</td>
</tr>
<tr>
<td>April 2015</td>
<td>15 February 2015</td>
<td>Resilient Systems</td>
<td>Katri Hakola, David Yarbrough, Paul Cleary, and Scott Jackson</td>
</tr>
<tr>
<td>July 2015</td>
<td>15 May 2015</td>
<td>Model-Based Systems Engineering</td>
<td>Joachim Fuchs</td>
</tr>
</tbody>
</table>