Software Engineering Institute

Building Assured Systems Framework

Nancy R. Mead
Julia H. Allen

September 2010

TECHNICAL REPORT
CMU/SEI-2010-TR-025
ESC-TR-2010-025

CERT® Program

http://www.sei.cmu.edu

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about SEI publications, please visit the library on the SEI website (http://www.sei.cmu.edu/library).

Table of Contents

List of Figures

List of Tables

Acknowledgments

This work relies heavily on the Master of Software Assurance Reference Curriculum (MSwA2010) [Mead 2010]. It would not have been possible to develop this document without the MSwA2010 body of knowledge and the support of our MSwA2010 coauthors Mark Ardis, Tom Hilburn, Andrew Kornecki, Rick Linger, and Jim McDonald. Another key document for us was the *2009 CERT Research Annual Report*, which provided us with current CERT research project information. We would like to thank John Goodenough and Carol Woody for their thoughtful review of this report. They made many valuable comments and suggestions for improvement. Thank you also to Rachel Callison on the SEI Library staff who helped us with references. We are appreciative of all the support we have received while working on this effort and document.

Executive Summary

This report's authors initiated the research described in this report in response to observing that there is no single, recognized framework to organize research topics and areas of practice focused on building assured systems (BAS). A building assured systems framework (BASF) could provide a meaningful context and structure within which to describe, compare, and contrast research and development methods for building assured systems. It could also be used to identify gaps, prioritize new research projects, and stop or decommission current research projects that are not contributing useful results as defined by the framework.

We began this inquiry by addressing these "pain points" raised by the CERT® Program's customers and sponsors and use these as our research questions:

- How do I decide which security methods fit into a specific life-cycle activity?
- How do I know if a specific security method is sufficiently mature for me to use on my projects?
- When should I take a chance on a security research approach that has not been widely used?
- What actions can I take when I have no approach or method for prioritizing and selecting new BAS research, or when promising research appears to be unrelated to other research in the field?

In the CERT Program at Carnegie Mellon University's Software Engineering Institute (SEI), such a framework can also be used to demonstrate how CERT research efforts and results contribute to building assured systems. Areas related to software assurance, such as software safety, reliability, and dependability, are not our primary focus, although we recognize that these areas provide important contributions to software assurance. Information assurance, as distinct from software assurance, was also not our primary focus at this time, although the protection of information in deployed software is important and could be considered in more depth in the future.

To understand previous and current work that could inform BASF development, we started by examining a number of existing software development and acquisition life-cycle process models, models for the development of more secure software, and research frameworks in software security and assurance. Summary descriptions of those that we reviewed appear in the first three sections of this report. Descriptive material often appears as direct excerpts from the source models and frameworks (noted as sans-serif, blue type). These can be easily skimmed if desired.

With this information, we formed a hypothesis that the recently developed Master of Software Assurance (MSwA2010) body of knowledge (BoK) [Mead 2010] could serve as our starting point for the BASF. This makes sense given that the curriculum BoK draws extensively from more than 25 sources describing methods, practices, and technologies for software assurance and security (including the software security models considered in this report). Also, as the authors of this report, we led and contributed to the development of the MSwA2010 curriculum.

® CERT is a registered mark owned by Carnegie Mellon University.

We tested this hypothesis by assigning "maturity levels"[1] to each area of the MSwA2010 BoK. BoK areas include assurance across life cycles, risk management, assurance assessment, assurance management, system security assurance, system functionality assurance, and system operational assurance. We defined these levels as follows:

- L1—The area provides guidance for how to think about a topic for which there is no proven or widely accepted approach. The intent of the area is to raise awareness and aid the reader in thinking about the problem and candidate solutions. The area may also describe promising research results that may have been demonstrated in a constrained setting.
- L2—The area describes practices that are in early pilot use and are demonstrating some successful results.
- L3—The area describes practices that have been successfully deployed (mature) but are in limited use in industry or government organizations. They may be more broadly deployed in a particular market sector.
- L4—The area describes practices that have been successfully deployed and are in widespread use. Readers can start using these practices today with confidence. Experience reports and case studies are typically available.

To test this hypothesis further, we mapped existing CERT research work, described in the *2009 CERT Research Annual Report* [CERT 2010], to the MSwA2010 BoK to see whether there are corresponding BoK areas for each research project. All major research projects did correspond to one or more BoK areas, either directly or indirectly. This gave us confidence that the BoK areas (and the research from which they were derived) could be used as our initial framework. Once we mapped the current CERT research projects to the MSwA2010 BoK, we performed an initial gap analysis to identify some promising research areas for CERT.

The BASF helps to address some, but not all, of the four research questions stated previously. Since the BASF naturally covers the development life cycle, mapping a particular security method to the appropriate knowledge area(s) does help to answer the first question (relationship of security method to life-cycle phase). For the second question (security method maturity), considering knowledge area maturity levels in conjunction with examining a specific method provides information to help decide whether the method is sufficiently mature for use. The third question is a bit harder to answer and requires more work on the part of a BASF user. A cost/benefit analysis or risk assessment aids in answering the third question of whether it is worth taking a chance on a method that has not been widely used.

From a research perspective, researchers could consider periodically rating the maturity of their methods using an approach such as that described above. This would assist BASF users in deciding which methods to use. It would also be helpful if researchers and research methods users could begin to collect and provide cost/benefit data. All too often, researchers and research method users decide on a particular method but do not collect any information to determine whether the benefit justified the cost or to help inform future decisions.

[1] The development and definition of these maturity levels support our work in software security engineering [Allen 2008].

We believe the BASF provides a context and structure for CERT's research work in building assured systems and that it can be used to show how various research efforts fit together. The gap analysis that we have done could be used to help in selecting new research and, to some extent, in prioritizing research projects. We anticipate that the BASF could be used in planning and justifying CERT's research program and communicating about it with others.

We expect that the U.S. Department of Defense (DoD) and other sponsors will find the BASF useful for tracking current research and development (R&D) efforts in building assured systems and possibly in acquiring assured systems.

Abstract

Researchers at the CERT® Program, part of Carnegie Mellon University's Software Engineering Institute, need a framework to organize research and practice areas focused on building assured systems. The Building Assured Systems Framework (BASF) addresses the customer and researcher challenges of selecting security methods and research approaches for building assured systems. After reviewing existing life-cycle process models, security models, and security research frameworks, the authors used the Master of Software Assurance Reference Curriculum knowledge areas as the BASF. The authors mapped all major CERT research areas to the BASF, proving that the BASF is useful for organizing building assured systems research. The authors also performed a gap analysis to identify promising CERT research areas. The BASF is a useful structure for planning and communicating about CERT research. The BASF will also be useful to CERT sponsors to track current research and development efforts in building assured systems.

1 The Problem[2]

There is no single, recognized framework to organize research and practice areas focused on building assured systems (BAS). Sponsors of the CERT® Program's research at the Carnegie Mellon® Software Engineering Institute (SEI) could use such a framework to help address the following challenges, including customer "pain points" and general research problems:

- How do I decide which security methods fit into a specific life-cycle activity?
- How do I know if a specific security method is sufficiently mature for me to use on my projects?
- When should I take a chance on a security research approach that has not been widely used?
- What actions can I take when I have no approach or method for prioritizing and selecting new research or when promising research appears to be unrelated to other research in the field?

Although the pain points use the term "security," the terms "security" and "assurance" are often used interchangeably when it comes to building systems. Our work relates to the following definition of software assurance from *Software Assurance Curriculum Project Volume I: Master of Software Assurance Reference Curriculum* [Mead 2010]:

> Application of technologies and processes to achieve a required level of confidence that software systems and services function in the intended manner, are free from accidental or intentional vulnerabilities, provide security capabilities appropriate to the threat environment, and recover from intrusions and failures.

Areas related to software assurance, such as software safety, reliability, and dependability, are not our primary focus, although we recognize that these areas provide important contributions to software assurance. Information assurance, as distinct from software assurance, was also not our primary focus at this time, although the protection of information in deployed software is important and could be considered in a follow on effort.

We define a framework using the following definitions from Babylon dictionary [Babylon 2009]:

> A framework is a basic conceptual structure used to solve or address complex issues. This very broad definition has allowed the term to be used as a buzzword, especially in a software context.
>
> A structure to hold together or support something, a basic structure.

The Building Assured Systems Framework (BASF) provides a meaningful context and structure within which to describe research and development for building assured systems. For example, the framework could be used in CERT to demonstrate how CERT research efforts contribute to building assured systems.

[2] The authors also address this problem in the *2009 CERT Research Annual Report* [CERT 2010].

® CERT is a registered mark owned by Carnegie Mellon University.

Background on Assured Systems

The following topics from internal research planning[3] at CERT in some way address the problem of BAS. These topics exhibit varying levels of maturity and use differing terminology, but they all are involved in building assured systems:

- engineering resilient systems—encompasses secure software engineering, as well as requirements engineering, architecture and design of secure systems and large systems of systems, and service and system continuity of operations
- containment—focuses on the problem of how to monitor and detect a component's behavior to contain and isolate the effect of aberrant behavior while still being able to recover from a false assumption of bad behavior
- architecting secure systems—defines the necessary and appropriate design artifacts, quality attributes, and appropriate tradeoff considerations that describe how security properties are positioned, how they relate to the overall system/IT architecture, and how security quality attributes are measured
- secure software engineering (secure coding, software engineering, and hardware design improvement)—improves the way software and hardware are developed by reducing vulnerabilities from software and hardware flaws. This work includes technology life-cycle assurance mechanisms, advanced engineering disciplines, standards and certification regimes, and best practices. Research areas in secure software engineering include refining current assurance mechanisms and developing new ones where necessary, developing certification regimes, and exploring policy and incentive options.

Secure software engineering encompasses a range of activities targeting security. *Software Security Engineering* presents a valuable discussion of these topics [Allen 2008]. In varying levels of detail, the book examines the spectrum of these appropriate activities:

- requirements engineering for secure software
- secure architecture and design
- secure coding and testing
- security and complexity: system assembly challenges
- governance and management for more secure software

Although these topics are discussed in *Software Security Engineering* [Allen 2008] additional research is ongoing. In fact, several of these topics are the focus of current research projects in CERT: security requirements engineering, secure coding, governance and management, software security measurement and analysis, and systems complexity, including global software supply chain, distributed management environments, and systems of systems.

Some organizations have begun to pay more attention to BAS, including

- organizations participating in the Building Security In Maturity Model (currently 30) [McGraw 2010]
- Microsoft's software development lifecycle (SDL) [Lipner 2005]

[3] These materials are not publicly available.

- Software Assurance Forum for Excellence in Code (SAFECode) consortium members [SAFECode 2010]
- Oracle
- members of the Open Web Application Security Project (OWASP) using the Software Assurance Maturity Model (SAMM)

These efforts tend to be stronger in software product development organizations, which characterize the type of organizations that have provided the most significant contribution to the efforts listed above. However, they are weaker in large organizations that develop systems for use in house and integrate systems across multiple vendors. They are also weaker in small- to medium-sized organizations developing software products for licensed use. Furthermore, there are a variety of life-cycle models in practice—no single approach has emerged as standard. Even in the larger organizations that adopt secure software engineering practices, there is a tendency to select a subset of the total set of recommended or applicable practices. Such uneven adoption of practices for BAS makes it difficult to evaluate the results using these practices.

Approach

We, the authors of this report, started BASF research by reviewing existing frameworks and life-cycle models for BAS. In the literature, we typically see life-cycle models or approaches that serve as structured repositories of practices from which organizations select those that are meaningful for their development projects. Some of these are discussed in *Software Security Engineering* [Allen 2008], as well as in the article "Software [In]security: A Software Security Framework: Working Towards a Realistic Maturity Model" [McGraw 2008].

Summary descriptions of several software development and acquisition process models that are in active use appear in Section 2, models for software security are summarized in Section 3, and descriptions of applicable security research frameworks appear in Section 4. Readers who are less interested in this background information can skim it or skip ahead to Section 5. In Section 5, we explore the hypothesis that the recently developed Master of Software Assurance (MSwA2010) body of knowledge (BoK) [Mead 2010] could serve as our starting point for BASF. The curriculum body of knowledge draws extensively from more than 25 sources describing methods, practices, and technologies for software security (including the models described in Section 3). In Section 6, we describe how current related research activities within CERT fit into the proposed BASF as a proof of concept for how BASF can be used. We then identify BASF as a framework that could serve as a context and structure for research into how to build assured systems (Section 7) and identify gap areas in Section 8. In Section 9 we conclude the report and outline future work to extend the results. This report could be used in the CERT Program's research planning and communications with others and also in the *CERT Research Annual Report*. We expect that the U.S. Department of Defense (DoD) and other sponsors would find it useful for tracking current research and development (R&D) efforts in BAS and possibly in acquiring assured systems.

2 Process Models for Software Development and Acquisition

A framework for building assured systems needs to build upon and reflect known, accepted, common practice for software development and acquisition. One commonly accepted expression of the codification of effective software development and acquisition practices is a process model. Process models define a set of processes that, when implemented, demonstrably improve the quality of the software that is developed or acquired using such processes. The SEI has been a recognized thought leader for more than 20 years in developing capability and maturity models for defining and improving the process by which software is developed and acquired. This includes building a community of practitioners and reflecting their experiences and feedback in successive versions of the models. These models reflect commonly known good practices that have been observed, measured, and assessed by hundreds of organizations. Such practices serve as the foundation for building assured systems; it makes no sense to attempt to integrate software security practices into a software development process or life cycle if this development process is not defined, implemented, and regularly improved. Thus, these development and acquisition models serve as the basis against which models and practices for software security are considered. These development and acquisition models also serve as the basis for considering the use of promising research results. The models described in this section apply to newly developed software, acquired software, and extending the useful life of legacy software.

The content in this section is excerpted from publicly available SEI websites and reports. It summarizes the objectives of Capability Maturity Model Integration (CMMI®) models in general, CMMI for Development, and CMMI for Acquisition. Readers of this report should be familiar with software development and acquisition process models in general (including CMMI-based models) to better understand how software security practices, necessary for building assured systems, are implemented and deployed. In addition, reflecting the current body of common practice in BASF helps researchers to identify new areas of research and possible gaps versus revisiting topics that are already well defined with demonstrated solutions.

2.1 CMMI Models in General

According to the SEI's CMMI website [SEI 2010a]:[4]

Capability Maturity Model Integration (CMMI®) is a process improvement approach that provides organizations with the essential elements of effective processes that ultimately improve their performance. CMMI can be used to guide process improvement across a project, a division, or an entire organization. It helps integrate traditionally separate organizational functions, set process improvement goals and priorities, provide guidance for quality processes, and provide a point of reference for appraising current processes.

The benefits you can expect from using CMMI include the following:

- Your organization's activities are explicitly linked to your business objectives.

® CMMI is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

[4] Excerpted material is presented in this report in sans-serif, blue type.

- Your visibility into the organization's activities is increased to help you ensure that your product or service meets the customer's expectations.
- You learn from new areas of best practice (e.g., measurement and risk).

CMMI models are collections of best practices that you can compare to your organization's best practices and use to guide improvement to your processes.

2.1.1 CMMI for Software Development (CMMI-DEV)

The SEI's CMMI for Development website states the following [SEI 2010b]:

Whether your business is developing high-tech systems, consumer software, or IT services, you want to ensure the highest quality product or service reaches your customer on time. Using CMMI-DEV[5] as part of a process improvement program in your development organization can help you achieve on-time delivery and high quality, especially if your product or service relies heavily on software.

CMMI-DEV is used for process improvement in development organizations. CMMI-DEV is a model or collection of "best practices" that organizations follow to dramatically improve the effectiveness, efficiency, and quality of their product and service development work. CMMI-DEV also is supported by training courses and appraisal methodologies to help organizations objectively measure their improvement progress.

CMMI-DEV guidance covers the lifecycles of products and services from conception through delivery and maintenance. CMMI-DEV best practices are flexible enough to apply to a variety of industries, yet stable and consistent enough to provide a benchmark against which your organization can measure and compare itself.

Adopting CMMI-DEV is a solid, high-return investment that your organization can make to ensure long-term results. The business benefits experienced by organizations using CMMI-DEV in their process improvement programs include the following:

- better customer satisfaction
- increased quality
- more accurate schedules
- lower development costs
- substantial return on investment
- improved employee morale and reduced turnover

CMMI-DEV-based process improvement includes identifying your organization's process strengths and weaknesses and making process changes to turn weaknesses into strengths. CMMI-DEV best practices and process improvement goals are organized into intuitive groups called "process areas." Your organization chooses its path to excellence by focusing on the process areas most important to its business objectives.

CMMI-DEV includes the following 23 process areas, grouped into four categories [CMMI Product Team 2006]:

- **Process Management**: Process Management process areas contain the cross-project activities related to defining, planning, deploying, implementing, monitoring, controlling, appraising, measuring, and improving processes.
  - Organizational Process Focus

[5] Details can be found in *CMMI for Development, Version 1.2* [CMMI Product Team 2006] and at http://www.sei.cmu.edu/cmmi/tools/dev/.

  - Organizational Process Definition + IPPD[6] [integrated product and process development]
  - Organizational Training
  - Organizational Process Performance
  - Organizational Innovation and Deployment
- **Project Management:** Project Management process areas cover the project management activities related to planning, monitoring, and controlling the project.
  - Project Planning
  - Project Monitoring and Control
  - Supplier Agreement Management
  - Integrated Project Management +IPPD[7]
  - Risk Management
  - Quantitative Project Management
- **Engineering**: Engineering process areas cover the development and maintenance activities that are shared across engineering disciplines. The Engineering process areas were written using general engineering terminology so that any technical discipline involved in the product development process (e.g., software engineering or mechanical engineering) can use them for process improvement. The Engineering process areas also integrate the processes associated with different engineering disciplines into a single product development process, supporting a product-oriented process improvement strategy. Such a strategy targets essential business objectives rather than specific technical disciplines. This approach to processes effectively avoids the tendency toward an organizational "stovepipe" mentality. The Engineering process areas apply to the development of any product or service in the development domain (e.g., software products, hardware products, services, or processes).
  - Requirements Development
  - Requirements Management
  - Technical Solution
  - Product Integration
  - Verification
  - Validation
- **Support**: Support process areas cover the activities that support product development and maintenance. The Support process areas address processes that are used in the context of performing other processes. In general, the Support process areas address processes that are targeted toward the project and may address processes that apply more generally to the organization. For example, Process and Product Quality Assurance can be used with all the process areas to provide an objective evaluation of the processes and work products described in all the process areas.
  - Configuration Management
  - Process and Product Quality Assurance

[6] Organizational Process Definition (OPD) has one goal that applies only when using CMMI with the IPPD group of additions.

[7] Integrated Project Management (IPM) has one goal that applies only when using CMMI with the IPPD group of additions.

- Measurement and Analysis
- Decision Analysis and Resolution
- Causal Analysis and Resolution

2.1.2 CMMI for Acquisition (CMMI-ACQ)

The SEI's CMMI for Acquisition website states the following [SEI 2010c]:

CMMI–ACQ[8] is a CMMI model designed for use in managing a supply chain by those who acquire, procure, or otherwise select and purchase products and services for business purposes.

CMMI-ACQ provides guidance to acquisition organizations for initiating and managing the acquisition of products and services that meet the needs of the customer. The model focuses on acquirer processes and integrates bodies of knowledge that are essential for successful acquisitions.

CMMI-ACQ provides an opportunity for acquisition organizations to

- avoid or eliminate barriers and problems in the acquisition process through improved operational efficiencies
- initiate and manage a process for acquiring products and services, including solicitations, supplier sourcing, supplier agreement development and award, and supplier capability management
- utilize a common language for both acquirers and suppliers so that quality solutions are delivered more quickly and at a lower cost with the most appropriate technology

CMMI-ACQ and CMMI-DEV have many similarities and complement each other. As CMMI-ACQ is used by the acquirer, CMMI-DEV may be used by the supplier. The terminology, structure, and many practices are shared by these two models.

CMMI-ACQ has 22 process areas: six are specific to acquisition practices and 16 are shared with other CMMI models.

The six process areas that are specific to acquisition practices are

- Acquisition Requirements Development
- Solicitation and Supplier Agreement Development
- Agreement Management
- Acquisition Technical Management
- Acquisition Verification
- Acquisition Validation

Additionally, the model includes guidance on

- acquisition strategy
- typical supplier deliverables
- transition to operations and support
- integrated teams

[8] More information can be found in *CMMI® for Acquisition, Version 1.2* [CMMI Product Team 2007] and at http://www.sei.cmu.edu/cmmi/tools/acq/.

The 16 shared process areas include practices for project management, organizational process management, and infrastructure and support.

CMMI models are one foundation for well-managed and -defined software development and acquisition processes. The next section describes leading models and frameworks that define processes and practices for software security. Such processes and practices are, in large part, in common use by a growing body of organizations that are developing software to be more secure. We have yet to see significant use of these processes and practices in the acquisition community.

3 Software Security Frameworks, Models, and Roadmaps

In addition to considering process models for software development and acquisition, a framework for building assured systems needs to build upon and reflect known, accepted, common practice for software security. There are a growing number of promising frameworks and models for building more secure software. For example, Microsoft has defined their security development lifecycle (SDL) and made it publicly available. In their recently released version 2, the authors of the Building Security In Maturity Model (McGraw, Chess, and Migues) have collected and analyzed software security practices in 30 organizations.

In this section, we summarize seven models, frameworks, and roadmaps, excerpting descriptive information from publicly available websites and reports. This section summarizes the objectives and content of each effort. Readers should have a broad understanding of these models and their processes and practices to appreciate the current state of the practice in building secure software and to aid in identifying promising research opportunities to fill gaps.

3.1 Building Security In Maturity Model (BSIMM)

The Building Security In Maturity Model (BSIMM) introduction states the following [McGraw 2010]:

> The Building Security In Maturity Model (BSIMM) is designed to help an organization understand, measure, and plan a software security initiative. The BSIMM was created through a process of understanding and analyzing real-world data from nine leading software security initiatives and then validated and adjusted with data from twenty-one additional leading software security initiatives. Altogether, the BSIMM collectively represents the wisdom and knowledge of thirty firms with active and successful software security initiatives. Though particular methodologies differ, such as the Open Web Application Security Project (OWASP) Comprehensive, Lightweight Application Security Process (CLASP), the Microsoft SDL, or the Cigital Touchpoints, many initiatives share common ground. This common ground is captured and described in the BSIMM.
>
> BSIMM is appropriate for an organization whose overall business goals for software security include:
>
> - informed risk management decisions
> - clarity on what is “the right thing to do” for everyone involved in software security
> - cost reduction through standard, repeatable processes
> - increased code quality
>
> BSIMM is not a complete “how to” guide for software security, nor is it a one-size-fits-all model. Instead, BSIMM is a collection of good practices and activities that are in use today.

A maturity model is appropriate for building more secure software (a key component of building assured systems) because improving software security means changing the way an organization develops software, over time. The BSIMM provides a way to assess the state of an organization, prioritize changes, and demonstrate progress. Not all organizations need to achieve the same security goals, but all organizations can be measured with the same yardstick.

The BSIMM is meant to be used by those who create and execute a software security initiative. Most successful initiatives are run by a senior executive who reports to the highest levels in the organization, such as the board of directors or the chief information officer. These executives lead an internal group that BSIMM calls the Software Security Group (SSG), charged with directly executing or facilitating the activities described in the BSIMM. The BSIMM is written with the SSG and SSG leadership in mind.

Roles that are addressed in the BSIMM include:

- executive leaders, including active sponsor
- SSG (software security staff with deep coding, design, and architectural experience)
- builders, testers, and operations staff
- administrators
- line of business owners
- product managers

As an organizing structure for the body of observed practices in 30 organizations, the BSIMM uses a Software Security Framework (SSF) (see Table 1).

Table 1: BSIMM Software Security Framework [McGraw 2010]

Governance **Goal: Transparency, Accountability, Checks and Balances**	**Intelligence** **Goal: Auditability, Stewardship, Standardization**	**SSDL* Touchpoints** **Goal: Quality Control**	**Deployment** **Goal: Quality Control, Change Management**
Strategy & Metrics: planning; assigning roles and responsibilities; identifying software security goals; determining budgets; identifying metrics and gates	Attack Models: capture information to think like an attacker; threat modeling; abuse case development and refinement; data classification; attack patterns	Architecture Analysis: capture software architecture; apply risks and threats; adopt an architecture review process; build assessment and remediation plan	Penetration Testing: test for vulnerability in final configuration; provide input to defect management and mitigation
Compliance & Policy: identify controls for compliance; develop contractual controls (service level agreements) for externally developed software; set software security policy; audit against policy	Security Features & Design: create usable security patterns for major security controls; building middleware frameworks for controls; creating and documenting security guidance	Code Review: use code review tools; develop customized rules; develop profiles for tool use by role; perform manual analysis; track/measure results	Software Environment: operating system (OS) and platform patching; web application firewalls; installation and configuration documentation; application monitoring; change management; code signing
Training: awareness training; new hire training; SSG office hours; build social network; role-based training; provide specific information on error root causes; annual refresher; on-demand; training for vendors/external parties	Standards & Requirements: elicit security requirements; determine commercial, off-the-shelf (COTS); building standards for major security controls; creating security standards; creating a standards review board	Security Testing: integrate security into standard quality assurance (QA) processes; black box testing; fuzz testing; risk-driven white box testing; apply attack models; code coverage analysis; focus on vulnerabilities in construction	Configuration Management & Vulnerability Management: patch and update applications; version control; defect tracking and remediation; incident handling

* Software Security Development Lifecycle

3.2 CMMI Assurance Process Reference Model

The Department of Homeland Security (DHS) Software Assurance (SwA) Processes and Practices Working Group developed a draft process reference model (PRM) for assurance in July 2008 [DHS 2008]. This PRM recommends additions to CMMI-DEV v1.2 to address software assurance. The "assurance thread" description[9] includes Figure 1, which may be useful for addressing the life-cycle phase aspect of the BASF.

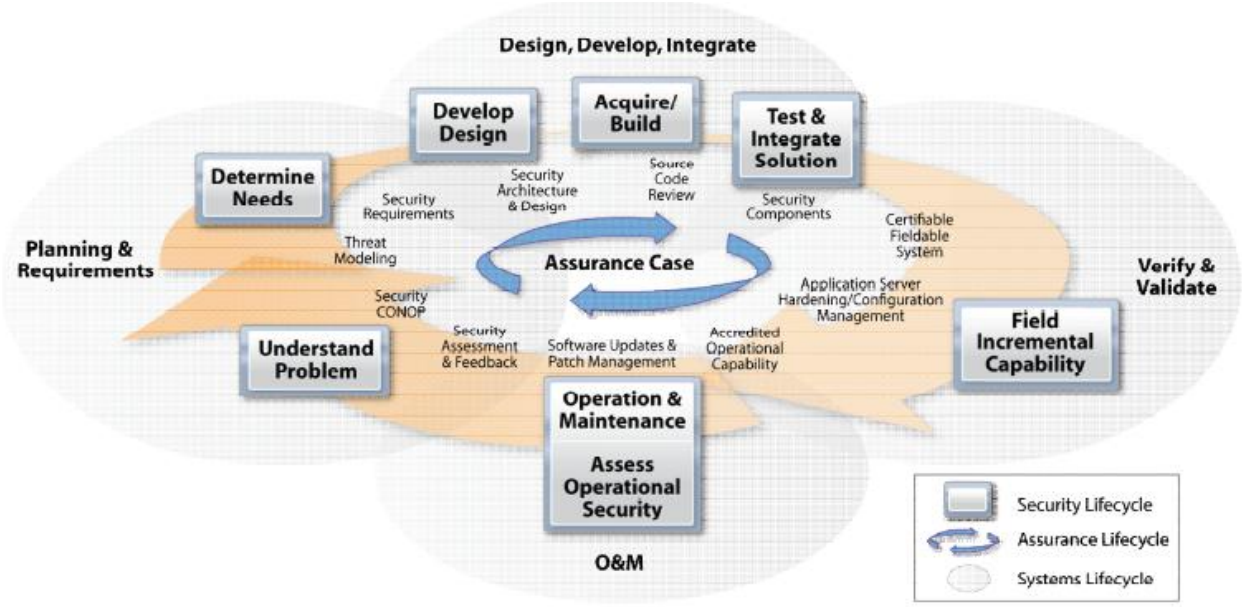

Figure 1: Summary of Assurance for CMMI Efforts

The DHS SwA Processes and Practices Working Group's additions and updates to CMMI-DEV v1.2 are focused at the specific practices (SP) level for the following CMMI-DEV Process Areas (PAs):

- Process Management
  - Organizational Process Focus
  - Organizational Process Definition
  - Organizational Training
- Project Management
  - Project Planning
  - Project Monitoring and Control
  - Supplier Agreement Management
  - Integrated Project Management
  - Risk Management
- Engineering
  - Requirements Development

[9] Available at https://buildsecurityin.us-cert.gov/swa/procwg.html.

  - Technical Solution
  - Verification
  - Validation
- Support
- Measurement & Analysis

3.3 Open Web Application Security Project (OWASP) Software Assurance Maturity Model (SAMM)

The following discussion of OWASP SAMM is from *Software Assurance Maturity Model (SAMM) v1.0* [OWASP 2009].

The Software Assurance Maturity Model (SAMM) is an open framework to help organizations formulate and implement a strategy for software security that is tailored to the specific risks facing the organization. The resources provided by SAMM will aid in:

- Evaluating an organization's existing software security practices
- Building a balanced software security assurance program in well-defined iterations
- Demonstrating concrete improvements to a security assurance program
- Defining and measuring security-related activities throughout an organization

SAMM was defined with flexibility in mind such that it can be utilized by small, medium, and large organizations using any style of development. Additionally, this model can be applied organization-wide, for a single line-of-business, or even for an individual project. Beyond these traits, SAMM was built on the following principles:

- *An organization's behavior changes slowly over time*—A successful software security program should be specified in small iterations that deliver tangible assurance gains while incrementally working toward long-term goals.
- *There is no single recipe that works for all organizations*—A software security framework must be flexible and allow organizations to tailor their choices based on their risk tolerance and the way in which they build and use software.
- *Guidance related to security activities must be prescriptive*—All the steps in building and assessing an assurance program should be simple, well-defined, and measurable. This model also provides roadmap templates for common types of organizations.

The foundation of the model is built upon the core business functions of software development with security practices tied to each [see Table 2]. The building blocks of the model are the three maturity levels defined for each of the twelve security practices. These define a wide variety of activities in which an organization could engage to reduce security risks and increase software assurance. Additional details are included to measure successful activity performance, understand the associated assurance benefits, estimate personnel and other costs.

Table 2: OWASP SAMM Business Functions and Security Practices [OWASP 2009]

Governance	Construction	Verification	Deployment
Strategy & Metrics: overall strategic direction of the software assurance program & instrumentation of processes & activities to collect metrics about an organization's security posture	Threat Assessment: identify and characterize potential attacks on software to better understand the risks and facilitate risk management	Design Review: inspect artifacts created from the design process to ensure provision of adequate security mechanisms and adherence to expectations for security	Vulnerability Mgmt: establish consistent process for managing internal and external vulnerability reports to limit exposure and gather data to enhance the security assurance program
Policy & Compliance: set up a security and compliance control and audit framework to achieve increased assurance in software under construction and in operation	Security Requirements: promote the inclusion of security-related requirements during the software development process to specify correct functionality from inception	Code Review: assess source code to aid vulnerability discovery and related mitigation activities as well as establish a baseline for secure coding expectations	Environment Hardening: implement controls for the operating environment in which software executes to bolster the security posture of applications that have been deployed
Education & Guidance: increase security knowledge amongst personnel in software development through training and guidance on security topics relevant to individual job functions	Secure Architecture: bolster the design process with activities to promote secure-by-default designs and control over technologies and frameworks upon which software is built	Security Testing: test software in its runtime environment in order to discover vulnerabilities and establish a minimum standard for software releases	Operational Enablement: identify and capture security-relevant information needed by an operator to properly configure, deploy, and run software

In *Software Assurance Maturity Model (SAMM) v1.0,* success metrics are presented for all activities in all 12 practices for all four critical business functions. Each practice has three objectives; each objective has 2 activities, for a total of 72 activities.

3.4 DHS SwA Measurement Work by Bartol and Moss

According to the DHS SwA Measurement Working Group [DHS 2010a]

Practical Measurement Framework for Software Assurance and Information Security[10] provides an approach for measuring the effectiveness of achieving software assurance goals and objectives at an organizational, program, or project level. It addresses how to assess the degree of assurance provided by software, using quantitative and qualitative methodologies and techniques. This framework incorporates existing measurement methodologies and is intended to help organizations and projects integrate SwA measurement into their existing programs.

The following discussion is excerpted from the *Practical Measurement Framework for Software Assurance and Information Security* [Bartol 2008].

Software assurance is interdisciplinary and relies on methods and techniques produced by other disciplines, including project management, process improvement, quality assurance, training, information security/information assurance, system engineering, safety, test and evaluation, software acquisition, reliability, and dependability [as shown in Figure 2].

[10] Bartol, Nadya. *Practical Measurement Framework for Software Assurance and Information Security, Version 1.0*. Practical Software & Systems Measurement (PSM). http://www.psmsc.com/Prod_TechPapers.asp (2008).

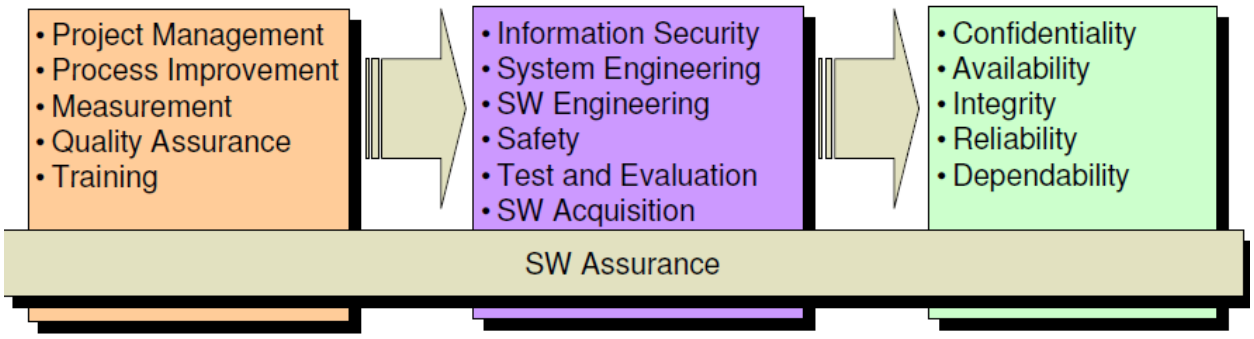

Figure 2: Cross-Disciplinary Nature of SwA [Bartol 2008]

The Practical Measurement Framework focuses principally, though not exclusively, on the information security viewpoint of SwA. Many of the contributing disciplines of SwA enjoy an established process improvement and measurement body of knowledge, such as quality assurance, project management, process improvement, and safety. SwA measurement can leverage measurement methods and techniques that are already established in those disciplines, and adapt them to SwA. The Practical Measurement Framework report focuses on information assurance/information security aspects of SwA to help mature that aspect of SwA measurement.

This framework provides an integrated measurement approach, which leverages five existing industry approaches that use similar processes to develop and implement measurement as follows:

- Draft National Institute of Standards and Technology (NIST) Special Publication (SP) 800-55, Revision 1, *Performance Measurement Guide for Information Security*
- ISO/IEC 27004 Information technology – Security techniques - Information security management measurement
- ISO/IEC 15939, *System and Software Engineering - Measurement Process*, also known as Practical Software and System Measurement (PSM)
- CMMI Measurement and Analysis Process Area
- CMMI GQ(I)M – Capability Maturity Model Integration Goal Question Indicator Measure

The Practical Measurement Framework authors selected these methodologies because of their widespread use among the software and systems development community and the information security community. The Framework includes a common measure specification table which is a crosswalk of specifications, templates, forms and other means of documenting individual measures provided by the five industry approaches listed above that were leveraged to create the framework.

Measures are intended to help answer the following five questions:

- What are the defects in the design and code that have a potential to be exploited?
- Where are they?
- How did they get there?
- Have they been mitigated?
- How can they be avoided in the future?

A number of representative key measures for different stakeholder groups are included in the framework to help organizations assess the state of their SwA efforts during any stage of a project:

- Supplier – an individual or an organization that offers software and system-related products and services to other organizations. This includes software developers, program managers, and other staff working for an organization that develops and supplies software to other organizations.
- Acquirer – an individual or an organization that acquires software and system-related products and services from other organizations. This includes acquisition officials, program managers, system integrators, system owners, information owners, operators, designated approving authorities (DAAs), certifying authorities, independent verification and validation (IV&V), and other individuals who are working for an organization that is acquiring software from other organizations.
- Within each supplier and acquirer organization, the following stakeholders are considered:
  - Executive Decision Maker – a leader who has authority to make decisions and may require quantifiable information to understand the level of risk associated with software to support decision-making processes.
  - Practitioner – an individual responsible for implementing SwA as a part of their job.

The framework describes candidate goals and information needs for each stakeholder group. The framework then presents example supplier measures as a table, with project activity, measures, information needs, and benefits as the columns. The rows include project activities—requirements management (5 measures), design (3 measures), development (6 measures), test (9 measures)—and the entire software development life cycle (SDLC) (3 measures).

Example measures for acquirers are similarly presented, intended to answer the questions

- Have SwA activities been adequately integrated into the organization's acquisition process?
- Have SwA considerations been integrated into the SDLC and resulting product by the supplier?

The acquisition activities (presented as rows) are planning (2 measures), contracting (3 measures), and implementation and acceptance (5 measures).

Ten example measures for executives are presented. These are intended to answer the question, "Is the risk generated by software acceptable to the organization?" Some of these are

- number and percent of patches published on announced date
- time elapsed for supplier to fix defects
- number of known defects by type and impact
- cost to correct vulnerabilities in operations
- cost of fixing defects before system becomes operational
- cost of individual data breaches
- cost of SwA practices throughout the SDLC

Fifteen example measures for practitioners are presented. These are intended to answer the question, "How well are current SwA processes and techniques mitigating software-related risks?"

3.5 Microsoft Security Development Lifecycle (SDL)

The Microsoft Security Development Lifecycle (SDL)[11] is an industry-leading software security process. A Microsoft-wide initiative and a mandatory policy since 2004, the SDL has played a critical role in embedding security and privacy in Microsoft software and culture. Combining a holistic and practical approach, the SDL introduces security and privacy early and throughout all phases of the development process.

The reliable delivery of more secure software requires a comprehensive process, so Microsoft defined *Secure by Design, Secure by Default, Secure in Deployment, and Communications* (SD3+C) to help determine where security efforts are needed. The guiding principles for SD3+C are identified in the following subsections, which are excerpted from *Microsoft Security Development Lifecycle Version 5.0* [Microsoft 2010b]:

Secure by Design

- **Secure architecture, design, and structure.** Developers consider security issues part of the basic architectural design of software development. They review detailed designs for possible security issues, and they design and develop mitigations for all threats.
- **Threat modeling and mitigation.** Threat models are created, and threat mitigations are present in all design and functional specifications.
- **Elimination of vulnerabilities.** No known security vulnerabilities that would present a significant risk to the anticipated use of the software remain in the code after review. This review includes the use of analysis and testing tools to eliminate classes of vulnerabilities.
- **Improvements in security.** Less secure legacy protocols and code are deprecated, and, where possible, users are provided with secure alternatives that are consistent with industry standards.

Secure by Default

- **Least privilege.** All components run with the fewest possible permissions.
- **Defense in depth.** Components do not rely on a single threat mitigation solution that leaves users exposed if it fails.
- **Conservative default settings.** The development team is aware of the attack surface for the product and minimizes it in the default configuration.
- **Avoidance of risky default changes.** Applications do not make any default changes to the operating system or security settings that reduce security for the host computer. In some cases, such as for security products, it is acceptable for a software program to strengthen (increase) security settings for the host computer. The most common violations of this principle are games that either open firewall ports without informing the user or instruct users to open firewall ports without informing users of possible risks.

[11] More information is available in *The Security Development Lifecycle* [Howard 2006], at the Microsoft Security Development Lifecycle website [Microsoft 2010a], and in the document *Microsoft Security Development Lifecycle Version 5.0* [Microsoft 2010b].

- **Less commonly used services off by default.** If fewer than 80 percent of a program's users use a feature, that feature should not be activated by default. Measuring 80 percent usage in a product is often difficult because programs are designed for many different personas. It can be useful to consider whether a feature addresses a core/primary use scenario for all personas. If it does, the feature is sometimes referred to as a P1 feature.

Secure in Deployment

- **Deployment guides.** Prescriptive deployment guides outline how to deploy each feature of a program securely, including providing users with information that enables them to assess the security risk of activating non-default options (and thereby increasing the attack surface).
- **Analysis and management tools.** Security analysis and management tools enable administrators to determine and configure the optimal security level for a software release.
- **Patch deployment tools.** Deployment tools aid in patch deployment.

Communications

- **Security response.** Development teams respond promptly to reports of security vulnerabilities and communicate information about security updates.
- **Community engagement.** Development teams proactively engage with users to answer questions about security vulnerabilities, security updates, or changes in the security landscape.

The secure software development process model with the addition of elements of SD3+C looks like the one shown in Figure 3.

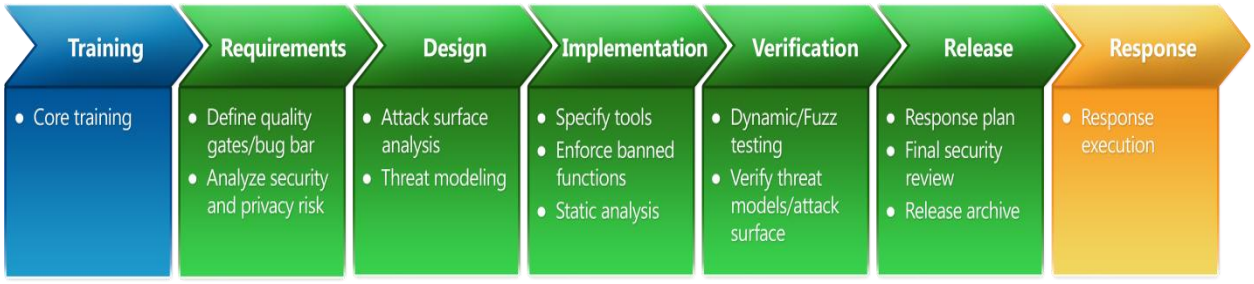

Figure 3: Secure Software Development Process Model at Microsoft [Microsoft 2010a]

The Microsoft SDL documentation describes, in great detail, what architects, designers, developers, and testers are required to do during each life-cycle phase.

The introduction states, "Secure software development has three elements—best practices, process improvements, and metrics. This document focuses primarily on the first two elements, and metrics are derived from measuring how they are applied" [Microsoft 2010b]. This infers that there is no concrete measurement-related information in this document; measures would need to be derived from each of the life-cycle-phase practice areas.

3.6 CERT® Resilience Management Model Resilient Technical Solution Engineering Process Area

As is the case for software security and software assurance, resilience is a property of software and systems. Developing and acquiring resilient[12] software and systems requires a dedicated process focused on this property that encompasses the software and system life cycle. As described in the CERT® Resilience Management Model's (CERT®-RMM)[13] Resilient Technical Solution Engineering (RTSE) process area,[14] the process defines what is required to develop resilient software and systems and is as follows [Caralli 2010]:

- Establish a plan for addressing resiliency as part of the organization's (or supplier's) regular development life cycle and integrate the plan into the organization's corresponding development process. Plan development and execution includes identifying and mitigating risks to the success of the project.
- Identify practice-based guidelines that apply to all phases such as threat analysis and modeling as well as those that apply to a specific life cycle phase.
- Elicit, identify, develop, and validate assurance and resiliency requirements (using methods for representing attacker and defender perspectives, for example). Such processes, methods, and tools are performed alongside similar processes for functional requirements.
- Use architectures as the basis for design that reflect a resiliency and assurance focus, including security, sustainability, and operations controls.
- Develop assured and resilient software and systems through processes that include secure coding of software, software defect detection and removal, and the development of resiliency and assurance controls based on design specifications.
- Test assurance and resiliency controls for software and systems and refer issues back to the design and development cycle for resolution.
- Conduct reviews throughout the development life cycle to ensure that resiliency (as one aspect of assurance) is kept in the forefront and given adequate attention and consideration.
- Perform system-specific continuity planning and integrate related service continuity plans to ensure that software, systems, hardware, networks, telecommunications, and other technical assets that depend on one another are sustainable.
- Perform a post-implementation review of deployed systems to ensure that resiliency (as well as assurance) requirements are being satisfied as intended.
- In operations, monitor software and systems to determine if there is variability that could indicate the effects of threats or vulnerabilities and to ensure that controls are functioning properly.
- Implement configuration management and change control processes to ensure software and systems are kept up to date to address newly discovered vulnerabilities and

---

[12] There is substantial overlap in the definitions of assured software (or software assurance) and resilient software (or software resilience). Resilient software is software that continues to operate as intended (including recovering to a known operational state) in the face of a disruptive event (satisfying business continuity requirements) so as to satisfy its confidentiality, availability, and integrity requirements (reflecting operational and security requirements). [Caralli 2010]

[13] See http://www.cert.org/resilience/.

[14] The RTSE document [Caralli 2010] can be downloaded from http://www.cert.org/resilience/rmm.html.

weaknesses (particularly in vendor-acquired products and components) and to prevent the intentional or inadvertent introduction of malicious code or other exploitable vulnerabilities.

Table 3 lists RTSE practices.

Table 3: RTSE Practices

Goals	Practices
RTSE:SG1 Establish Guidelines for Resilient Technical Solution Development	RTSE:SG1.SP1 Identify General Guidelines
	RTSE:SG1.SP2 Identify Requirements Guidelines
	RTSE:SG1.SP3 Identify Architecture and Design Guidelines
	RTSE:SG1.SP4 Identify Implementation Guidelines
	RTSE:SG1.SP5 Identify Assembly and Integration Guidelines
RTSE:SG2 Develop Resilient Technical Solution Development Plans	RTSE:SG2.SP1 Select and Tailor Resiliency Guidelines
	RTSE:SG2.SP2 Integrate Selected Guidelines with a Defined Software and System Development Process
RTSE:SG3 Execute the Plan	RTSE:SG3.SP1 Monitor Execution of the Development Plan
	RTSE:SG3.SP2 Release Resilient Technical Solutions into Production

In addition to RTSE, the following are goals and practices in other CERT-RMM process areas that organizations should consider when developing and acquiring software and systems that need to meet assurance and resiliency requirements [Caralli 2010]:

- Resiliency requirements for software and system technology assets in operation, including those that may influence quality attribute requirements in the development process, are developed and managed in the Resiliency Requirements Development (RRD) and Resiliency Requirements Management (RRM) process areas respectively.
- Identifying and adding newly developed and acquired software and system assets to the organization's asset inventory is addressed in the Asset Definition and Management (ADM) process area.
- The management of resiliency for technology assets as a whole, particularly for deployed, operational assets, is addressed in the Technology Management (TM) process area. This includes, for example, asset fail-over, backup, recovery, and restoration.
- Acquiring software and systems from external entities and ensuring that such assets meet their resiliency requirements throughout the asset life cycle is addressed in the External Dependencies Management process area. That said, RTSE specific goals and practices should be used to aid in evaluating and selecting external entities that are developing software and systems (EXD:SG3.SP3), formalizing relationships with such external entities (EXD:SG3.SP4), and managing an external entity's performance when developing software and systems (EXD:SG4).
- Monitoring for events, incidents, and vulnerabilities that may affect software and systems in operation is addressed in the Monitoring (MON) process area.
- Service continuity plans are identified and created in the Service Continuity (SC) process area. These plans may be inclusive of software and systems that support the services for which planning is performed.

RTSE assumes that the organization has one or more existing, defined processes for software and system development into which resiliency controls and activities can be integrated. If this is not

the case, the organization should not attempt to implement the goals and practices identified in RTSE or in other CERT-RMM process areas as described above.

3.7 International Process Research Consortium (IPRC) Roadmap

From August 2004 to December 2006, the SEI's process program sponsored a research consortium of 28 international thought leaders to explore process needs for today, the foreseeable future, and the unforeseeable future. One of the emerging research themes was the relationships between processes and product qualities, defined as "understanding if and how particular process characteristics can affect desired product (and service) qualities such as security, usability, and maintainability" [IPRC 2006]. As an example or "instantiation" of this research theme, Allen and Kitchenham (two of the participating members) developed research nodes and research questions for security as a product quality. This content helps identify research topics and gaps that could be explored within the context of the BASF.

The descriptive material presented in Table 4 is excerpted from *A Process Research Framework* [IPRC 2006].

Table 4: IPRC Research Nodes and Questions for Security as a Product Quality

Research Node	Research Questions
Establishing security in the systems or software development life cycle: Determine the extent to which processes can be used to accurately reflect and cause the instantiation of required security product quality attributes for each software development life-cycle (SDLC) phase.	How is security expressed in each phase of the SDLC? What are appropriate expressions, from a security perspective, of how the system is to be used?
	What processes best ensure the instantiation of established security principles?
	What are effective processes and methods that ensure that known causes of security vulnerabilities are not present in each phase of the SDLC?
	What processes and methods can be used to accelerate adoption of known methods for developing low-defect-rate (and thus more secure) software? (state of art/state of practice gap)
	What are the compelling cost/benefit arguments to do so?
	Is it possible to build and verify secure software and systems using agile methods?
	What processes can be used to ensure that security requirements are met for systems composed from existing components? For extensible systems?
Establishing the relationship between process and security as a product quality: Establish whether there is a direct relationship between security as product quality and the processes used to develop the product.	What is the role of process in ensuring that software and systems are engineered such that they continue to function correctly under malicious attack, failure, and accidents?
Measuring and monitoring security performance: Establish processes to accurately capture meaningful measures that aid in determining if a system is meeting its security requirements (and how well) during all SDLC phases.	What are the definitions of meaningful, informative security measures? What processes are needed to reliably collect these?
	What measures indicate that a system has met its security requirements for each SDLC phase? What are the processes for collecting, analyzing, and reporting these measures?

Research Node	Research Questions
	What measures and evaluation processes can be used to determine the effectiveness of different secure software development processes?
Verification and validation of security: Enable managers to select appropriate assessment, evaluation, verification, and validation processes to confirm the achievement of security requirements. Process selection is guided by the nature and complexity of the system being constructed and operated. Methods include the use of scenario-based misuse/abuse cases.	How is an adequate or acceptable level of security determined, tested, verified, and certified?
	What processes are most effective for assessing, evaluating, verifying, and certifying the security of software and systems (including those provided by third parties)?
	What processes and methods are most likely to reveal security issues, flaws, and vulnerabilities during each SDLC phase? And with third party, open source, and COTS, or other component software? In the case where such processes already exist and have empirical evidence to justify their use, what can be done to accelerate their adoption? (state of art/state of practice gap)
	What processes and methods allow for building misuse/abuse cases that predictably provide evidence that security product qualities are present?
Sustaining adequate security: Enable managers to select processes that result in establishing, sustaining, and evolving an adequate level of security throughout the full product life cycle.	How do we define and sustain adequate security in the face of increasingly sophisticated attacks (attack evolution), technology evolution, enterprise evolution, supply chain evolution, and the like (all sources of change that require a system to evolve)?
Usable security: Enable users to effectively apply and use required security mechanisms, to the extent these are visible to the user.	What user interface processes and methods result in users applying protection and security mechanisms routinely, automatically, and correctly?
	What processes result in minimal to no user involvement in security?
Using the marketplace to drive adequate security: Establish processes resulting in a consumer/customer marketplace that will not purchase software known to be insecure.	What processes, market forces, and other mechanisms can be used to require organizations that produce software with a significant annual volume of reported vulnerabilities to improve their products?

As a companion discussion to software development and software security models and frameworks, the next section of this report provides comparable information on research frameworks for security. Ideally, the BASF needs to reflect the best thinking from all of these domains.

4 Security Research Roadmaps, Agendas, and Frameworks

In researching this area, one of our first observations was that the terms "roadmaps," "agendas," and "frameworks" were often used interchangeably. Sometimes the term "framework" was used to refer to specific development frameworks. Other times the authors would use the term "framework" to refer to a research agenda. The term "roadmap" was usually used to designate a research agenda as well.

We identified a number of security research frameworks in the literature and in our own experience that could be good candidates for BASF. After surveying these frameworks, it became clear that some of them were geared towards specific topics, and hence not good candidates for the framework that would serve as an umbrella for a wide variety of software assurance research topics. We summarize the specialized frameworks briefly in section 4.1.

We then go on to discuss the broader research frameworks in more detail in section 4.2. Each of the broader frameworks has its own organization, which is reflected in the summaries and quotes that we provide. In some cases the broader frameworks include a conclusion provided by the framework authors. Note that conclusions that appear in these subsections are not our conclusions, but the conclusions of the original authors. In other cases, the authors provide a research agenda or roadmap. As a consequence, many of these frameworks could not be compared directly to one another.

4.1 Security Research Frameworks for Specific Topics

In the process of identifying candidate security research frameworks that could be applied to our definition of software assurance, we came across a number of frameworks related to security, but they often were very specific and generally related more to development than to research. In some cases, the term "frameworks" was used synonymously with international standards. In other cases, frameworks were used to support architectural decisions, implementation models, and coding standards. Examples of such frameworks include the following:

- Quality Attribute Reasoning Frameworks. In *Security and Survivability Reasoning Frameworks and Architectural Design Tactics*, the authors discuss security and survivability reasoning frameworks in conjunction with architectural tactics. The authors state, "Our approach includes a collection of 'quality attribute reasoning frameworks' that understand both quality attribute reasoning and how architects design for the quality attribute under particular situations" [Ellison 2004]. An example is the use of reasoning frameworks for inhibiting denial-of-service (DDoS) attacks.
- Architectural Frameworks for Composable Survivability and Security. This was a Defense Advanced Research Projects Agency (DARPA) project led by Peter Neumann from 2001 to 2004. It included three tasks: distributed systems and network architectures with three subtasks of composability, design principles, and architecture; consultation by SRI International with related projects; and a short-term effort using a static analysis approach. Information is available at http://www.csl.sri.com/users/neumann/chats.html.

- *Secure Software Development through Coding Conventions and Frameworks*. In this paper, the authors state that, "If a framework is able to provide automatic sanitizing for all kinds of commands that the specification requires, it will be an ideal *security framework*" [Okubo 2007]. They go on to propose specific classes as a security framework.
- *A Framework for Composable Security Definition, Assurance, and Enforcement*. The author says, "My doctoral research proposes a composable security definition, assurance, and enforcement via a model-driven framework that preserves separation of security concerns from modeling through implementation, and provides mechanisms to compose these concerns into the application while maintaining consistency between design models and code" [Pavlich-Mariscal 2006].This is clearly a development framework, rather than a research framework.
- *Desperately Seeking Security Frameworks—A Roadmap for State CIOs*. This white paper discusses a variety of standards such as ISO 27001 as frameworks [NASCIO 2009]. In this context, a framework is a standard intended to assist in auditing and compliance.
- *JavaScript Hijacking—Only 1 Out of 12 Popular AJAX Frameworks Prevents It*. In this article, the authors discuss frameworks that consist of AJAX toolkits and libraries [O'Neil 2008]. Although interesting, this is clearly a discussion of implementation-level frameworks.

4.2 Broad Security Frameworks for Research

When we narrowed the field to frameworks that support research in assured software, we found the following candidates, although they often pointed to research agendas rather than providing a more general research framework that could be used to support current and future research to support building assured software.

4.2.1 ICSE 2000 Software Engineering for Security: A Roadmap by Devanbu and Stubblebine

"Software Engineering for Security: A Roadmap" presents research issues that arise in the interactions between software engineering and security [Devanbu 2000]. It is organized so that the topics parallel a Waterfall life-cycle model. In each topic area, the authors survey current work in the field and the challenges. The challenges point towards an agenda for future research. The topics, excerpted from various sections of this report, are

Requirements and Policies

- security models and policies
  - challenge: unifying security with systems engineering
  - challenge: unifying security and system models

Architecture and Design of Secure Systems

- re-engineering for security
  - challenge: legacy security mismatches
  - challenge: separating the security "aspect"

Software Piracy & Protection

- adversary economics

- approaches to protection: hardware and software tokens, dynamic decryption of code, watermarking, code partitioning
- challenge: attacker cost models

Trusting Software Components

- black box approaches, grey box approaches, cryptographic coverage verification, tamper-resistant hardware
  - challenges: more grey box approaches

Verification of Systems

- challenge: implementation-based verification methods

Secure Software Deployment

- secure configuration management
  - challenge: controlled delegation
  - privacy protection

Secure Computations, Not Secure Computers

Conclusions of Devanbu and Stubblebine

In "Software Engineering for Security: A Roadmap," the authors discuss the notion that systems are error-prone, and that a desirable goal would be to secure computations rather than systems. They discuss notions of correctness proofs and associated use of cryptography towards securing computations.

4.2.2 Observations on Information Security Crisis by Jussipekka Leiwo

The paper *Observations on Information Security Crisis* [Leiwo 1999] surveys the symptoms and causes of the information security crisis, and sketches an outline of an approach required for tackling the crisis. An excerpted brief outline of the main points of the paper follows.

1. Symptoms of the Crisis
   - software security problems
   - communication protocol security problems
   - problems with cryptographic primitives
2. Causes of the Symptoms
   - lack of mechanisms for evaluating security
   - a gap between management and enforcement of information security
   - conflicts between security and top-down system design principles
   - lack of support for information security in non-traditional organizations
   - lack of consensus on definitions of concepts involved
   - scientific challenges in information systems security research
3. Solutions for the Causes
   - research on the information security software crisis —Surprisingly, the concept of the information security software crisis has not been subjected to much detailed academic research.
   - research on the flexibility of security safeguards—This is one of the major still-unanswered research questions in information systems security. Flexibility

should be integrated in both security measures and mechanisms for specifying these measures.

- comprehensive security of information systems—This requires contributions from many scientific fields: theory of computability to justify and evaluate security measures, computer and communications security to establish a model of security, software engineering to adequately implement the security model, systems analysis and design to capture the nature of security requirements, and socio-ethical considerations to establish and enforce operational procedures and guidelines for information security. *To establish a scientific foundation for information systems security, existing frameworks from related disciplines need to be considered from the security point of view.*[15]
- Flexible safeguards and relationships need to be established between information systems research and information systems security research.
- Mechanisms that integrate the design of security and of systems in general need to be established.

Conclusions of Leiwo's Paper

Leiwo has studied a fundamental problem of system vulnerability. According to the author,

the cause of the problem lies in the weak scientific foundation for information systems security and relatively primitive system security design methods. Security violations suggest a lack of understanding of security concepts among both researchers and practitioners. Fundamentally different definitions originating from different subsets of the information systems community make it difficult to deal with information systems security in a comprehensive way. *To overcome these problems, generic research frameworks are needed to support information systems security*. Otherwise, research will remain fragmented and inconsistent, and security measures will continue to prevent both normal and innovative system operations.

4.2.3 Engineering Secure Complex Software Systems and Services by ERCIM

The Security and Trust Management Working Group of The European Research Consortium for Informatics and Mathematics (ERCIM) and the European Commission's Directorate General Information Society Unit F5 "Security" jointly organized a strategic seminar called Engineering Secure Complex Software Systems and Services [ERCIM 2008].

The seminar objective was to link academic and industrial expertise in secure software engineering with industry best practices. The specific objectives of the seminar were to

- present the best practices applied in industry and to discuss key R&D initiatives
- encourage dialogue and collaboration between research scientists and industrial players
- identify future research challenges, in particular in the context of the evolution towards the future of the internet

More than 60 stakeholders from industry and academia attended the seminar. Brief highlights from the three panels follow, which were taken and edited slightly from the complete report that is available at ERCIM's website [ERCIM 2008].

[15] Author's emphasis retained.

1. Industrial Best Practices and Perspectives

   - Best practices
     Major industrial players in private and public organizations have great interest in cooperating in this field by sharing and promoting pragmatic approaches and proven software assurance practices. Automated support for best practice enforcement and the ability to reason about the business impact of security are key issues that need to be addressed to manage security related efforts in an economically feasible way.
   - Novel IT frameworks, models and tools during all phases of the software lifecycle
     Software security should be an integral part of every phase of the software lifecycle. The existence of common IT development and execution frameworks enforces the use of best practices and fosters collaborative work towards further improvement in achieving higher levels of secure software. Industry requires tools that encapsulate specialized knowledge by translating underlying theoretical foundations into concrete secure software development practices.
   - Creating the business case for security
     IT security has to compete with several other industry investment priorities. With squeezing IT budgets and ever-shorter times to market, managers need to assess how much to spend on IT security. Understanding the value that investments on secure software can add through the product value chain is vital for business and IT managers making decisions on security expenditures. Specifically, managers need to understand how much risk their company is ready to take for a given threat and manage that risk accordingly.
   - Dealing with assurance, measurability and testing
     Understanding the value of security and assessing and managing risks implies putting in place an appropriate set of "controls" at different levels. Such a control framework would allow prevention of vulnerabilities and monitoring of compliance. That requires, however, an appropriate set of independent measurement and testing procedures for all phases of the software lifecycle as well as metrics for collecting data, auditing performance and, ultimately, proving/ensuring security by measuring it.
   - Dealing with increasing levels of complexity of software systems
     Complexity is rapidly increasing when moving from the secure engineering of isolated application components to that of "systems of systems," with functionality often different from what their underlying components were designed for. Moreover, they increasingly rely on real-time dynamic composition involving third-party software components and services. Under these circumstances, achieving secure systems and secure software products is a huge challenge and key business success factor.
   - Promoting education and awareness
     Security-conscious and well-educated software architects and developers are needed, along with more investment in education and training. The importance of secure software needs to be stressed among managers, software architects, programmers and users.

2. Research Advances and Perspectives

The second panel of the seminar focused on promising research directions for engineering secure complex software systems.

- Security requirements engineering

Security weaknesses originate in incomplete or conflicting software security requirements. Specific expertise, methods and tools should be devoted to security requirements engineering. For example, a step-by-step refinement procedure and automated tools would help security requirements engineers to improve the process from requirements elicitation to analysis and to track them during the subsequent software development steps.

- Models for Secure Software Engineering

  The software development process needs several models to deal with domain specific aspects and to identify the correct security solutions. These models often have to be combined and refined in a way that ensures that the overall security of the final product is kept. Composability is a major security challenge related to systems scalability and complexity. Another challenge, from a security viewpoint, is dynamic change of systems and code and dynamic evolution of system functionalities. The high cost of applying formal methods is an impediment to their larger industrial deployment. Therefore, one of the research directions with major impact would be to embed formal methods in automated development tools in a transparent way for the user. Finally, methods for measuring the trustworthiness of the software systems is yet another area of importance for industry where major research efforts are necessary.

- Language-based security

  Language-based security is regarded as the backbone of secure software engineering. Language-based security techniques and specific type systems move the burden of ensuring the security of the final code from the application programmer to the programming environment developers. A promising research area is developing techniques for proving complex properties of cryptographic algorithms as well as provably correct implementations.

- Advances in security verification and validation

  Several rigorous techniques have been developed for checking system specifications, such as model checking and theorem proving. However, there are still several limitations that must be addressed for their wider deployment in industry. Relevant research issues include addressing their scalability and coping with the ever-increasing complexity of software-intensive systems. More research effort is needed to make security verification and validation tools usable in practice.

- Advances in risk assessment for systems of systems

  Risk is a crucial notion in security and its role in the design of complex systems of systems needs to be further investigated. Embedding risk in an explicit manner in all the steps of the software development lifecycle could help to reduce the cost and make the improvements in software engineering more concrete.

3. The Way Forward

The last panel considered the findings from the two first panels and introduced additional issues related to: (a) enabling methodologies and tools for building secure complex systems and services; (b) software liability aspects; and (c) standardization, education, and other relevant issues for the field.

- Enabling methodologies and tools for building secure complex software systems

  Security engineering and software engineering methodologies and platforms should be integrated. The general (wrong) perception is that software engineering is dealing with construction of correct software, while security engineering is dealing with the deployment of software. Industry also needs usable and efficient methodologies and tools

that automate the security of software code. It is urgent to undertake further work for bridging the gap between fundamental theories and pragmatic approaches for industry to use. Software is often built on top of legacy systems and/or is outsourced. This calls for tools for verifying the security properties and performance of legacy systems and/or third party software. Composability is a big challenge. Even if a software system is built from individually trusted components, the overall system may not be trusted.

- Software liability

  Software companies in general and those companies in particular offering packaged software services or Service Oriented Architecture (SOA)-based applications and services are not liable for the damages they may cause due to software vulnerabilities of their products. As liability may change with time, it is important for companies to adopt best practices quickly. A prerequisite for solving software liability is solving the composability problem.

- Standardization, education and other relevant issues

  Currently there is a lack of sufficient standards in software security. In some cases, clear specifications are available at a certain level of abstraction, but implementations of standards are often not completely in line with these specifications. Robust tools for testing and validating such implementations are necessary. Often there is a gap between the methodologies that secure software engineers are taught in Universities and the knowledge they need when working in industry. More cooperation is required between industry and academia in order to produce curricula dealing with both foundational knowledge principles and industrial reality.

4. Concluding Remarks of the ERCIM Meeting

The participation of both industry and academia representatives at the event shows the importance of the topics addressed. Industry is motivated to adopt best practices in software security engineering, and the scientific community has methodologies and tools to offer. Targeting specific priorities identified in the report would help to close the gap between theoretical and practical work. *Security and software engineering also need to be integrated into a coherent framework.*[16] As systems complexity increases, easy-to-use software tools need to be developed through research and industrial partnerships. In order to ease this process, industry and academia should share expertise and adopt the same language and terminology.

Raising current levels of education and awareness in the field is another emerging theme. New forms of IT infrastructures such as cloud computing bring new challenges for secure software as well as new opportunities.

4.2.4 CERT Research Roadmap

In 2009 and 2010, under the leadership of Archie Andrews, CERT developed an internal research roadmap of potential topics of interest. The roadmap was built with input from a number of external seminar speakers as well as in-house sources. A research advisory group provided review and comments. Since the roadmap was an internal working paper, it did not have a concluding section. A summary of the topics and brief descriptions from the research roadmap follow:

1. Measures, Situational Awareness, and Response

   - Metric Repository. Understand what quantitative measures of effectiveness should be collected to provide a global perspective.

[16] Author's emphasis retained.

- Cybersecurity Metrics. Identify the critical measurements that describe, for an organization, if their security posture is at an adequate or acceptable level.
- Trend Analysis. Develop the approaches for gaining awareness of various types of message traffic across the Internet and sufficient understanding to measure vulnerabilities and attack mechanisms.
- Cyber Situational Intelligence and Response. Identify tools and techniques that provide greater awareness of the state of an IT environment resulting in a timely response to security factors.
- Intelligence Awareness and Assessment. Broaden the approach beyond reactive (attacks and countermeasures), technical (bytes, network interfaces, and protocols), and procedural (policies and standards) to understand cyber security implications of "front page news;" assess the cyber security dimension of geopolitical and economic trends.
- Active Cyber Defense. Investigate attribution, trace back, decision criteria for reaction (cut off or monitor attack)
- Governance. Identify what should be said in the board room and how to deliver the message to get required results.

2. Systems and Software Engineering
    - Engineering Resilient Systems. Address secure software engineering, including requirements engineering, architecture and design of secure systems, and large systems of systems.
    - Containment. Monitor and detect a component's behavior in such a manner as to contain and isolate the effect of aberrant behavior while still being able to recover from a false assumption of bad behavior.
    - Composable Systems. Understand the parameters required to address the security characteristics of modules and assemblages of modules that can be composed into systems in such a manner that the security characteristics of the composed systems are understood.
    - Best Secure System Engineering Practices. Define those systems engineering principles and practices necessary to build secure systems.
    - Architecting Secure Systems. Define the necessary and appropriate design artifacts, quality attributes, and appropriate tradeoff considerations that describe how security properties are positioned, how they relate to the overall system/IT architecture, and how security quality attributes are measured.
    - Secure Software Engineering. Improve the way software and hardware are developed to reduce vulnerabilities from software and hardware flaws to include technology life-cycle assurance mechanisms, advanced engineering disciplines, standards and certification regimes, and best practices.
    - Operating Legacy Systems. Understand how to provide secure systems that run the latest applications in a safe environment; requires preserving existing properties in a malevolent environment.
3. Security of Cooperating Objects
    - Security of Cooperating Objects. Understand and address the security requirements inherent in the emerging computing model of semi-autonomous heterogeneous cooperating entities creating a shared, unpredictable state.

- Secured Concurrent Processing. Understand how to ensure security in a multi-processor/multi-process environment where computing components must manage shared tasks and shared trust.

4. Control Systems
   - Designing Secure Control Systems. Improve the security of process control systems and associated information networks to include secure control systems architectures and necessary protocols to address standards for control system security.
   - Control System CERT/CC. Provide an incident tracking and response capability for control systems users and vendors.
5. Security Modeling, Simulation, and Testing
   - Testbed. Develop a security testbed, using the latest technologies such as authentication and access control techniques, etc.
   - Modeling and Testing. Identify or develop scalable simulator tools and test beds to understand the security state of currently deployed technologies, as well as the readiness of technologies about to be deployed in the field.
6. Special Topics
   - Transition CERT Developments to Adoption. Develop effective transition mechanisms for getting all that we have done and all that we plan to do more effectively adopted, and be willing to stick with these until we have some true measures of successful use (or not) in the field.
   - Intrinsic Internet Infrastructure Protocols Security. Improve security in foundational protocols and others on which the information infrastructure is built.
   - Forensics. Identify, track, and bring cybercriminals to justice.
   - Trust and Privacy. Identify ways to ensure that IT systems protect the privacy rights of individuals using IT systems while maintaining overall system security.
   - Data Capture History. Determine how to create a history of collection, change, and deletion at the appropriate level of granularity to produce an auditable sequence of assignable events.
   - Identity Management. Ensure access to resources based on the identity of the requestor.
   - Temporal Coherence. Associate uniform time value with data in order to properly order event sequence.
   - Mobile Security. Develop security considerations for a mobile workforce and operations to include devices, operating procedures, operational security, and planning for recovering potentially distributed information.
   - Electronic Balloting. Understand and address the issues surrounding an electronic voting and tabulating process.
   - Security in Social Computing. Assess the potential security and privacy problems associated with social interactions in a networked environment.
   - Security Implications of Climate Change and Environmental Sustainability. Evaluate the potential impact of looming climate changes on presently accepted practices.

4.2.5 Knowledge Transfer Network Roadmap

The Cyber Security Knowledge Transfer Network (KTN) held an invitation-only meeting in March 2009, attended by a number of international experts in software security. During the meeting a vision was laid out for "software and systems which are resilient and sustainable by design" [Jones 2009]. As with some of the other reports, there are no specific conclusions, but rather a roadmap pointing the way forward. The vision described in the KTN report that documents the meeting results is [Jones 2009]

The development and procurement of software and systems which are resilient and sustainable by design, where requirements such as security and privacy are, as a matter of course, defined at project initiation and implemented and assured throughout in risk-based, whole-life processes.

To support the vision, a roadmap was laid out in 5 areas. Some of these areas point towards future research, whereas others are focused on coordination and communications:

1. Environmental shaping
    - Any strategy must deal with the motivations and incentives that lead people to implement or ignore good engineering practices. These are described in points below.
    - Define cost-effective business models. If engineering good practices are to be adopted there must be a clear business benefit.
    - Establish procurement strategy, procedures and requirements. Consideration should be given to how procurement strategy and requirements can shape the behavior of suppliers.
    - Manage supply chain risk. Related to procurement, is the ongoing assessment of products and service delivered throughout the supply chain, ensuring that 'non-explicit security requirements' are derived and met by vendors and suppliers.
    - Establish legislative and regulatory framework. This would involve the development of a framework which formalizes the requirement to be diligent in the development of software, systems and services.
    - Nurture consumer demand. This involves tackling awareness amongst consumers and buyers of software and services so that they ask the right questions of suppliers and develop a company's competitive advantage by addressing customer needs.
2. Information exchange and concept development
    - Developers and designers need to be equipped with the concepts that will underpin their analysis, assessment and planning. This underpinning knowledge will also form the logic on which tools and technical services are developed.
    - Establish mechanisms for information exchange, encouraging transparency, open standards and innovation through combining the latest in digital media publication together with national and international communities of interest.
    - Develop a dynamic library of threats, vulnerabilities, attack patterns and risk models. Risk driven engineering practices will only be effective when addressing the latest threats and vulnerabilities.
    - Establish semantics for 'non-functional' requirements engineering. It is still not clear that there is a common understanding about how nonfunctional

requirements may be expressed or captured, let alone how they might be enacted by an engineer or designer.

- Determine whole-life development processes. While some organizations have developed their own 'best practice,' more work needs to be done on acceptance of what constitutes good practice and what might become a standard.
- Determine measurable assurance and validation approaches. Many security products and services are still offered on the basis of unsubstantiated claims. Further fundamental research in assurance in security and other metrics needs to be conducted.

3. Technical facilitation

There is no doubt that the management, monitoring, modeling, testing, verification and validation of complex software and systems needs technical facilitation:

- Utilize secure coding languages. Some commonly used languages (e.g. C, php) allow, or even encourage, programming practices that introduce security vulnerabilities.
- Develop modeling and analytical tools for planning and assessment. During development and design of systems, modeling capability will support functions such as predictive analysis and what-if planning.
- Establish trusted libraries of 'reusable code' and components. Reusable stocks of 'application blocks' and components which have been through significant testing and assurance may help generate trust and improve quality.
- Define interoperability standards for functionality and testing. In support of the development of technical standards for functionality and testing, much more work needs to be done in interoperability standards.
- Develop analysis and testing tools for deployed systems and systems of systems. The development of tools is particularly challenging yet necessary.

4. Professionalization

The professionalization process is an important contributor to institutionalizing ways of working, maintaining standards and spreading good practice.

- Establish the role of 'independent architect'. Make use of the services of an independent architect throughout the procurement and implementation process to provide independent advice and support.
- Develop national and international standards. The role of standards in spreading good practice and assurance is key.
- Design curricula for universities and colleges. Awareness, education and training were given a high priority throughout the Paris meeting with many citing the need to capture, document and share curricula.
- Update engineering accreditation core competencies. Review core competencies, perhaps even including an ethical dimension concerning software and application standards and fitness for purpose.
- Ensure professional bodies nurture good practice. Levels of certification, continuing professional development and agenda setting could be facilitated by professional bodies.

5. Communications strategy

A communications strategy layered across each of the lines of development above is essential for their full support and to generate the desired impact. The following describe a communications planning framework that can used to support delivery of the vision.

- Determine desired behaviors and attitudes of audiences. Through the development of concepts, good practice and standards, explicit statements of desired attitudes and behaviors should be developed.
- Select and analyze audiences. A clear understanding of what drives, motivates and interests each audience is necessary if communication is to be successful.
- Determine the message. Messages should be developed for each audience and shaped to resonate with their intended audience.
- Establish interactive communication channels. Channels where people interact are likely to generate more engagement than one-way documents, websites and posters.
- Monitor and evaluate communications strategy. Purposeful communications requires monitoring and evaluation if it is going to continue to be relevant and effective.

4.2.6 DHS Cyber Security Research Roadmap

In 2009, DHS published a research roadmap, identifying the following current hard problems in information security research [DHS 2009]:

1. Scalable trustworthy systems (including system architectures and requisite development methodology)
2. Enterprise-level metrics (including measures of overall system trustworthiness)
3. System evaluation life cycle (including approaches for sufficient assurance)
4. Combatting insider threats
5. Combatting malware and botnets
6. Global-scale identity management
7. Survivability of time-critical systems
8. Situational understanding and attack attribution
9. Provenance (relating to information, systems, and hardware)
10. Privacy-aware security
11. Usable security

4.2.7 Cyber Security Research and Development Agenda

Doug Maughan defines a cyber security research agenda in the "Inside Risks" column of the January 2010 *Communications of the ACM*. This agenda draws upon the DHS cyber security research agenda to highlight the following ten areas, taken from the article [Maughan 2010]:

1. Software Assurance: poorly written software is at the root of all of our security problems;
2. Metrics: we cannot measure our systems, thus we cannot manage them;
3. Usable Security: information security technologies have not been deployed because they are not easily usable;

4. Identity Management: the ability to know who you are communicating with will help eliminate many of today's online problems, including attribution;
5. Malware: today's problems continue because of a lack of dealing with malicious software and its perpetrators;
6. Insider Threat: one of the biggest threats to all sectors that has not been adequately addressed;
7. Hardware Security: today's computing systems can be improved with new thinking about the next generation of hardware built from the start with security in mind;
8. Data Provenance: data has the most value, yet we have no mechanisms to know what has happened to data from its inception;
9. Trustworthy Systems: current systems are unable to provide assurances of correct operation to include resiliency; and
10. Cyber Economics: we do not understand the economics behind cybersecurity for either the good guy or the bad guy.

4.3 Assessment of Security Research Frameworks

Although the security research frameworks in the literature were interesting, they tended to describe gaps in current practices and methods rather than provide a framework that would serve as an umbrella for existing research work as well as support gap analysis. We also found that some of these research frameworks were quite broad, including hardware, physical security, and operations. Other topics, while important to security in general, were not quite on target for building assured systems.

5 Indicators of Method Maturity and the MSwA2010 Body of Knowledge (BoK)

In parallel with our work on identifying a framework for building assured systems, we were developing a body of knowledge (BoK) to support a Master of Software Assurance Reference Curriculum (MSwA2010) [Mead 2010]. MSwA2010 was a year-long effort to identify a body of knowledge for software assurance that would support a Master of Software Assurance Reference Curriculum and, ultimately, Master of Software Assurance degree programs. The participants who developed MSwA2010, in addition to the coauthors of this report, included other SEI staff members and faculty members from Embry-Riddle Aeronautical University, Stevens Institute of Technology, and Monmouth University. The MSwA2010 report contains a discussion of prerequisites, outcomes, body of knowledge, curriculum architecture, and course descriptions, among other things.

The process of developing the MSwA2010 curriculum included an extensive study of SDLC practices used to build assured software and identification of associated references through literature searches and also based on the expertise of the curriculum authors. The practices were classified into practice categories, and from there we developed knowledge units, which became elements of the BoK. We also did an informal coverage analysis to ensure that all practice categories were covered by at least one knowledge unit. In some cases, we excluded practice areas that were out of scope, such as privacy. Although we did not achieve complete traceability, the coverage exercise gave us confidence that our BoK had not excluded important practice areas for building assured systems. The level of effort invested in the development of the BoK suggested to us that that we might be able to use the MSwA2010 body of knowledge (BoK) as our initial Building Assured Systems Framework.

We studied the available models, roadmaps, and frameworks in Sections 2 through 4 of this document, and in fact the models in Sections 2 and 3 informed the curriculum effort. Although we did not do a formal tradeoff analysis, the study of the material in the literature and our deep knowledge of the MSwA2010 BoK and its development process reinforced our decision to use it as the initial foundation for the BASF. A formal tradeoff analysis could be done as part of our future work in this area, although we think it is unlikely that the outcome will be different.

To test the hypothesis that the MSwA2010 BoK might serve as the foundation for the BASF, we assigned the following maturity levels to each element of the MSwA2010 BoK. We developed these maturity levels to support our work in software security engineering (refer to *Software Security Engineering* [Allen 2008]). The association of BoK elements and maturity levels was accomplished by evaluating the extent to which relevant sources, practices, curricula, and courseware exist for a particular BoK element and the extent to which the authors have observed the element in practice in organizations.

Maturity Levels

- L1—The area provides guidance for how to think about a topic for which there is no proven or widely accepted approach. The intent of the area is to raise awareness and aid the reader

in thinking about the problem and candidate solutions. The area may also describe promising research results that may have been demonstrated in a constrained setting.

- L2—The area describes practices that are in early pilot use and are demonstrating some successful results.
- L3—The area describes practices that have been successfully deployed (mature) but are in limited use in industry or government organizations. They may be more broadly deployed in a particular market sector.
- L4—The area describes practices that have been successfully deployed and are in widespread use. Readers can start using these practices today with confidence. Experience reports and case studies are typically available.

Maturity Levels Assigned to the MSwA2010 BoK

This section contains the MSwA2010 BoK, which includes expected graduate outcomes, with maturity levels and education levels assigned.

Outcomes

We expect each graduate to have achieved outcomes after completing a master's degree based on the MSwA2010 reference curriculum. The outcomes are defined in each section of the MSwA2010 BoK.

MSwA2010 BoK with Outcomes and Maturity Levels

We found that the current maturity of the material being proposed for delivery in MSwA2010 varied. For example, a student would be expected to learn material at all maturity levels. If the practice was not very mature, we would still expect the student to be able to master it and use it in an appropriate manner after completing an MSwA program. The following content comes from the *Master of Software Assurance Reference Curriculum* report [Mead 2010].

1. Assurance Across Life Cycles

Outcome: Graduates will have the ability to incorporate assurance technologies and methods into life-cycle processes and development models for new or evolutionary system development, and for system or service acquisition.

1.1. Software Life-Cycle Processes

1.1.1. New development [L4]

Processes associated with the full development of a software system

1.1.2. Integration, assembly, and deployment [L4]

Processes concerned with the final phases of the development of a new or modified software system

1.1.3. Operation and evolution [L4]

Processes that guide the operation of the software product and its change over time

1.1.4. Acquisition, supply, and service [L3]

Processes that support acquisition, supply, or service of a software system

1.2. Software Assurance Processes and Practices

1.2.1. Process and practice assessment [L3]

Methods, procedures, and tools used to assess assurance processes and practices

1.2.2. Software assurance integration into SDLC phases [L2/3]

Integration of assurance practices into typical life-cycle phases (for example, requirements engineering, architecture and design, coding, test, evolution, acquisition, and retirement)

2. Risk Management

Outcome: Graduates will have the ability to perform risk analysis and tradeoff assessment and to prioritize security measures.

2.1. Risk Management Concepts

2.1.1. Types and classification [L4]

Different classes of risks (for example, business, project, technical)

2.1.2. Probability, impact, severity [L4]

Basic elements of risk analysis

2.1.3. Models, processes, metrics [L4] [L3—metrics]

Models, process, and metrics used in risk management

2.2. Risk Management Process

2.2.1. Identification [L4]

Identification and classification of risks associated with a project

2.2.2. Analysis [L4]

Analysis of the likelihood, impact, and severity of each identified risk

2.2.3. Planning [L4]

Risk management plan covering risk avoidance and mitigation

2.2.4. Monitoring and management [L4]

Assessment and monitoring of risk occurrence and management of risk mitigation

2.3. Software Assurance Risk Management

2.3.1. Vulnerability and threat identification [L3]

Application of risk analysis techniques to vulnerability and threat risks

2.3.2. Analysis of software assurance risks [L3]

Analysis of risks for both new and existing systems

2.3.3. Software assurance risk mitigation [L3]

Plan for and mitigation of software assurance risks

2.3.4. Assessment of Software Assurance Processes and Practices [L2/3]

As part of risk avoidance and mitigation, assessment of the identification and use of appropriate software assurance processes and practices

3. Assurance Assessment

Outcome: Graduates will have the ability to analyze and validate the effectiveness of assurance operations and create auditable evidence of security measures.

3.1. Assurance Assessment Concepts

3.1.1. Baseline level of assurance; allowable tolerances, if quantitative [L1]

Establishment and specification of the required or desired level of assurance for a specific software application, set of applications, or a software-reliant system (and tolerance for same)

3.1.2. Assessment methods [L2/3]

Validation of security requirements

Risk analysis

Threat analysis

Vulnerability assessments and scans [L4]

Assurance evidence

Knowledge of how various methods (such as those above) can be used to determine if the software or system being assessed is sufficiently secure within tolerances

3.2. Measurement for Assessing Assurance

3.2.1. Product and process measures by life-cycle phase [L1/2]

Definition and development of key product and process measurements that can be used to validate the required level of software assurance appropriate to a given life-cycle phase

3.2.2. Other performance indicators that test for the baseline as defined in 3.1.1, by life-cycle phase [L1/2]

Definition and development of additional performance indicators that can be used to validate the required level of software assurance appropriate to a given life-cycle phase

3.2.3. Measurement processes and frameworks [L2/3]

Knowledge of range of software assurance measurement processes and frameworks and how these might be used to accomplish software assurance integration into SDLC phases

3.2.4. Business survivability and operational continuity [L2]

Definition and development of performance indicators that can specifically address the software/system's ability to meet business survivability and operational continuity requirements, to the extent the software affects these

3.3. Assurance Assessment Process (collect and report measures that demonstrate the baseline as defined in 3.1.1.)

3.3.1. Comparison of selected measurements to the established baseline [L3]

Analysis of key product and process measures and performance indicators to determine if they are within tolerance when compared to the defined baseline

3.3.2. Identification of out-of-tolerance variances [L3]

Identification of measures that are out of tolerance when compared to the defined baselines and ability to develop actions to reduce the variance

4. Assurance Management

Outcome: Graduates will have the ability to make a business case for software assurance, lead assurance efforts, understand standards, comply with regulations, plan for business continuity, and keep current in security technologies.

4.1. Making the Business Case for Assurance

4.1.1. Valuation and cost/benefit models, cost and loss avoidance, return on investment [L3]

Application of financially-based approaches, methods, models, and tools to develop and communicate compelling cost/benefit arguments in support of deploying software assurance practices

4.1.2. Risk analysis [L3]

Knowledge of how risk analysis can be used to develop cost/benefit arguments in support of deploying software assurance practices

4.1.3. Compliance justification [L3]

Knowledge of how compliance with laws, regulations, standards, and policies can be used to develop cost/benefit arguments in support of deploying software assurance practices

4.1.4. Business impact/needs analysis [L3]

Knowledge of how business impact and needs analysis can be used to develop cost/benefit arguments in support of deploying software assurance practices, specifically in support of business continuity and survivability

4.2. Managing Assurance

4.2.1. Project management across the life cycle [L3]

Knowledge of how to lead software and system assurance efforts as an extension of normal software development (and acquisition) project management skills

4.2.2. Integration of other knowledge units [L2/3]

Identification, analysis, and selection of software assurance practices from any knowledge units that are relevant for a specific software development or acquisition project

4.3. Compliance Considerations for Assurance

4.3.1. Laws and regulations [L3]

Knowledge of the extent to which selected laws and regulations are relevant for a specific software development or acquisition project, and how compliance might be demonstrated

4.3.2. Standards [L3]

Knowledge of the extent to which selected standards are relevant for a specific software development or acquisition project, and how compliance might be demonstrated

4.3.3. Policies [L2/3]

Knowledge of how to develop, deploy, and use organizational policies to accelerate the adoption of software assurance practices, and how compliance might be demonstrated

5. System Security Assurance

Outcome: Graduates will have the ability to incorporate effective security technologies and methods into new and existing systems.

5.1. For Newly Developed and Acquired Software for Diverse Systems

5.1.1. Security and safety aspects of computer-intensive critical infrastructure [I2]

Knowledge of safety and security risks associated with critical infrastructure systems such as found, for example, in banking and finance, energy production and distribution, telecommunications, and transportation systems

5.1.2. Potential attack methods [L3]

Knowledge of the variety of methods by which attackers can damage software or data associated with that software by exploiting weaknesses in the system design or implementation

5.1.3. Analysis of threats to software [L3]

Analysis of the threats to which software is most likely to be vulnerable in specific operating environments and domains

5.1.4. Methods of defense [L3]

Familiarity with appropriate countermeasures such as layers, access controls, privileges, intrusion detection, encryption, and code review checklists

5.2. For Diverse Operational (Existing) Systems

5.2.1. Historic and potential operational attack methods [L4]

Knowledge of and ability to duplicate the attacks that have been used to interfere with an application's or system's operations

5.2.2. Analysis of threats to operational environments [L3]

Analysis of the threats to which software is most likely to be vulnerable in specific operating environments and domains

5.2.3. Designing of and plan for access control, privileges, and authentication [L3]

Design of and plan for access control and authentication

5.2.4. Security methods for physical and personnel environments [L4]

Knowledge of how physical access restrictions, guards, background checks, and personnel monitoring can address risks

5.3. Ethics and Integrity in Creation, Acquisition, and Operation of Software Systems

5.3.1. Overview of ethics, code of ethics, and legal constraints [L4]

Knowledge of how people who are knowledgeable about attack and prevention methods are obligated to use their abilities, both legally and ethically, referencing the Software Engineering Code of Ethical and Professional Conduct [ACM 2009]

5.3.2. Computer attack case studies [L3]

Knowledge of the legal and ethical considerations involved in analyzing a variety of historical events and investigations

6. System Functionality Assurance

Outcome: Graduates will have the ability to verify new and existing software system functionality for conformance to requirements and to help reveal malicious content.

6.1. Assurance Technology

6.1.1. Technology evaluation [L3]

Evaluation of capabilities and limitations of technical environments, languages, and tools with respect to creating assured software functionality and security

6.1.2. Technology improvement [L3]

Recommendation of improvements in technology as necessary within project constraints

6.2. Assured Software Development

6.2.1. Development methods [L2/3]

Rigorous methods for system requirements, specification, architecture, design, implementation, verification, and testing to develop assured software

6.2.2. Quality attributes [L3—depends on the property]

Software quality attributes and how to achieve them

6.2.3. Maintenance methods [L3]

Assurance aspects of software maintenance and evolution

6.3. Assured software analytics

6.3.1. Systems analysis [L2 architectures; L3/4 networks, databases (identity management, access control)]

Analysis of system architectures, networks, and databases for assurance properties

6.3.2. Structural analysis [L3]

Structuring the logic of existing software to improve understandability and modifiability

6.3.3. Functional analysis [L2/3]

Reverse engineering of existing software to determine functionality and security properties

6.3.4. Analysis of methods and tools [L3]

Capabilities and limitations of methods and tools for software analysis

6.3.5. Testing for assurance [L3]

Evaluation of testing methods, plans, and results for assuring software

6.3.6. Assurance evidence [L2]

Development of auditable assurance evidence

6.4. Assurance in acquisition

6.4.1. Assurance of acquired software [L2]

Assurance of software acquired through supply chains,[17] vendors, and open sources, including developing requirements and assuring delivered functionality and security

6.4.2. Assurance of software services [L3]

Development of service level agreements for functionality and security with service providers and monitoring compliance

7. System Operational Assurance

Outcome: Graduates will have the ability to monitor and assess system operational security and respond to new threats.

7.1. Operational Procedures

7.1.1. Business objectives [L3]

Role of business objectives and strategic planning in system assurance

7.1.2. Assurance procedures [L3]

Creation of security policies and procedures for system operations

7.1.3. Assurance training [L4]

Selection of training for users and system administrative personnel in secure system operations

7.2. Operational Monitoring

7.2.1. Monitoring technology [L4]

Capabilities and limitations of monitoring technologies, and installation and configuration or acquisition of monitors and controls for systems, services, and personnel

[17] For more information about software security supply chain risk, download the SEI report *Evaluating and Mitigating Software Supply Chain Security Risks* [Ellison 2010].

7.2.2. Operational evaluation [L4]

Evaluation of operational monitoring results with respect to system and service functionality and security

7.2.3. Operational maintenance [L3]

Maintenance and evolution of operational systems while preserving assured functionality and security

7.2.4. Malware analysis [L2/3]

Evaluation of malicious content and application of countermeasures

7.3. System Control

7.3.1. Responses to adverse events [L3/4]

Plan for and execution of effective responses to operational system accidents, failures, and intrusions

7.3.2. Business survivability [L3]

Maintenance of business survivability and continuity of operations in adverse environments (See also Outcome 3, Assurance Assessment.)

The next section of this report identifies the relationship between the MSwA2010 BoK and current research work-in-progress as described in the *2009 CERT Annual Research Report* [CERT 2010]. The purpose of this exercise is to determine if the MSwA2010 BoK can serve as a useful framework for structuring and describing security-related research activities.

6 Mapping of CERT Research to the MSwA2010 BoK

We next mapped major existing CERT research projects described in the *2009 CERT Research Annual Report* [CERT 2010] to the MSwA2010 BoK to see whether BoK areas corresponded to each research project. This was needed to help us decide whether the MSwA2010 BoK would be adequate as the foundation for the BASF. All major research projects mapped to BoK areas—see Table 5. In this table, the first column is the name of the project from the research report. The second column is the name(s) of the author(s) of the project description in the research report. The authors are usually the principal investigators of that research project. The third column indicates the BoK areas that are relevant to the research. The fourth column indicates whether there is a direct or indirect relationship between the research project and the BoK.

Some projects had a clear and direct relationship to the BoK areas. These projects were typically in the area of software development and acquisition. Other projects could not be directly related to the BoK, but in all cases there was an indirect relationship. Projects that did not directly relate to the BoK were often advanced analysis projects, such as Finding Malicious Activity in Bulk DNS Data. This mapping gave us confidence that the BoK areas could be used as our initial framework. In some cases, as noted, the relationship between the CERT projects and the BoK was indirect, but as a framework, the BoK held up.

Table 5: 2009 CERT Research Annual Report Major Projects

Project Name	Author	Corresponding MSwA2010 BoK Areas	Related Directly to MSwA2010 Topics (Mostly Software Development & acquisition)
Applying Function Extraction (FX) Techniques to Reverse Engineer Virtual Machines	Mark Pleszkoch, Stacy Prowell, Cory F. Cohen, and Jeffrey S. Havrilla	6.3 Assured Software Analytics, 6.3.3 Functional analysis (reverse engineering)	Yes
A Probabilistic Population Study of the Conficker-C Botnet	Rhiannon Weaver	3.2 Measurement for Assessing Assurance (modeling and measurement), 5.1.2 Potential attack methods	No
Finding Malicious Activity in Bulk DNS Data	Ed Stoner	7.2 Operational Monitoring, 7.2.4 Malware analysis	No
Function Extraction for Malicious Code Analysis	Kirk Sayre, Mark Pleszkoch, Timothy Daly, Richard Linger, and Stacey Prowell	7.2.4 Malware analysis	Yes

Project Name	Author	Corresponding MSwA2010 BoK Areas	Related Directly to MSwA2010 Topics (Mostly Software Development & acquisition)
Function Hashing for Malicious Code Analysis	Cory F. Cohen and Jeffrey S. Havrilla	5.1.2 Potential attack methods, 5.2.1 Historic and potential operational attack methods, 6.3 Assured Software Analytics, 6.3.3 Functional analysis (reverse engineering), 7.2 Operational Monitoring, 7.2.4 Malware analysis	Yes
Catching IPv6 Tunneled in IPv4	Evan Wright	6.3.1 Systems analysis (network analysis), 7.2.4 Malware Analysis (protocols, analysis)	No
Modeling Insider Theft of Intellectual Property	Andrew Moore, Dawn Cappelli, and Randy Trzeciak	2.1 Risk Management Concepts, 2.2 Risk Management Process, 2.3.1 Vulnerability and threat identification, 3.1.2 Assessment Methods (threat and vulnerability analysis), 5. System Security Assurance, 5.1.2 Potential attack methods, 5.1.3 Analysis of threats to software, 5.2 For Diverse Operational (Existing) Systems, 5.3.2 Computer attack case studies	Yes
Metrics for Evaluating Network Sensor Placement	Soumyo D. Moitra and Evan Wright	3.2.4 Business survivability and operational continuity (operational measurement), 7.2.1 Monitoring technology, 7.2.2 Operational evaluation	No
Rayon: A Unified Framework for Data Visualization	Philip Groce	7.2.1 Monitoring technology, 7.2.2 Operational evaluation	No
Source Code Analysis Laboratory	Robert Seacord, David Svoboda, and Philip Miller	6.2, 6.2.1, 6.3.1 (Secure coding, analysis)	Yes
SQUARE: Requirements Engineering for Improved System Security	Nancy R. Mead and Justin Zahn	1. Software Life-Cycle Processes,1.1.1 New development, 1.1.4 Acquisition, supply, and service, 2.2 Risk Management Process, 2.3 Software Assurance Risk Management, 6.2 Assured Software Development, 6.2.1 Development Methods	Yes

We did not have enough information about the additional newer research projects described in the *2009 CERT Research Annual Report*, nor were they far enough along to assess whether the projects related directly or indirectly to software development and acquisition. However, we did assign MSwA2010 BoK areas to those projects—see Table 6.

Table 6: 2009 CERT Research Annual Report Short Projects

Project Name	Authors	Corresponding MSwA2010 BoK Areas
Advanced Technology for Test and Evaluation of Embedded Systems	Timothy Daly and Richard Linger	6.2.1 Development methods (test), 6.3.4 Analysis of methods and tools, 6.3.5 Testing for assurance
Automatic Generation of Hidden Markov Models for the Detection of Polymorphic and Metamorphic Malware	Mark Pleszkoch, Cory F. Cohen, and Timothy Daly	5.1.2 Potential attack methods, 5.1.3 Analysis of threats to software, 7.2.4 Malware analysis
Baselining Port-Specific Scanning Behavior	Rhiannon Weaver	7.2.1 Monitoring technology, 7.2.2 Operational evaluation
Building Assured Systems Framework (BASF)	Nancy R. Mead and Julia Allen	1.2.2 Software assurance integration into SDLC phases, 2.3.4 Assessment of software assurance processes and practices, 4.2.2 Integration of other knowledge units
Control System Security and Critical Infrastructure Survivability	Howard F. Lipson	5.1.1 Security and safety aspects of computer-intensive critical infrastructure
Cyber Assurance	Christopher Alberts, Robert J. Ellison, and Carol Woody	3.1.2 Assessment methods, 4.1 Making the Business Case for Assurance
Cyber Security Risk Assessment in the Bulk Electric System	Samuel A. Merrell and James F. Stevens	3.1.2 Assessment methods, 5.1.1 Security and safety aspects of computer-intensive critical infrastructure
Influencing National Capability Development in Cyber Security through Incentives	Bradford Willke and Samuel A. Merrell	4.1 Making the Business Case for Assurance
Measuring Operational Resilience	Julia Allen	3.2.4 Business survivability and operational continuity, all of 7 System Operational Assurance
Measuring Software Security	Julia Allen	3.2 Measurement for Assessing Assurance, 3.3 Assurance Assessment Process
SiLK: Improvements and Plans	Mark Thomas and Michael Duggan	6.3.1 Systems analysis, 7.2.1 monitoring technology, 7.2.2 Operational evaluation
The Smart Grid Maturity Model	James F. Stevens and David W. White	1.1.3 Operation and evolution, 1.1.4 Acquisition, supply, and service, 1.2 Software Assurance Processes and Practices, 3.2 Measurement for Assessing Assurance, 3.3 Assurance Assessment Process, 4.1.4 Business impact/needs analysis, 4.3 Compliance Considerations for Assurance, 6.4 Assurance in Acquisition, all of 7 System Operational Assurance

Once this mapping was complete, we felt comfortable with the selection of the MSwA2010 BoK as the initial BASF. While the mapping was not perfect, in that certain research work was related

only indirectly to the BoK, we felt comfortable using it. The BoK reflected our collective understanding of what was needed to build assured systems, so in effect we had come full circle.

7 BASF Description

For the BASF, therefore, we will use the MSwA2010 BoK areas. However, in order to use it as a framework, we will not need to retain the MSwA2010 outcomes or brief descriptions shown in Section 5. This is because the outcomes and brief descriptions relate to the use of the BoK areas in the curriculum and are not germane to the use of the topic areas as an umbrella for our research. On the other hand, we developed the maturity levels specifically for the BASF and used them retrospectively in the curriculum, so the maturity levels are retained. In fact, we will use the maturity levels as an initial litmus test for whether research is needed in a specific area. Presumably we want to focus our research on topic areas that are less mature.

1. Assurance Across Life Cycles

1.1. Software Life-Cycle Processes

1.1.1. New development [L4]

1.1.2. Integration, assembly, and deployment [L4]

1.1.3. Operation and evolution [L4]

1.1.4. Acquisition, supply, and service [L3]

1.2. Software Assurance Processes and Practices

1.2.1. Process and practice assessment [L3]

1.2.2. Software assurance integration into SDLC phases [L2/3]

2. Risk Management

2.1. Risk Management Concepts

2.1.1. Types and classification [L4]

2.1.2. Probability, impact, severity [L4]

2.1.3. Models, processes, metrics [L4] [L3—metrics]

2.2. Risk Management Process

2.2.1. Identification [L4]

2.2.2. Analysis [L4]

2.2.3. Planning [L4]

2.2.4. Monitoring and management [L4]

2.3. Software Assurance Risk Management

2.3.1. Vulnerability and threat identification [L3]

2.3.2. Analysis of software assurance risks [L3]

2.3.3. Software assurance risk mitigation [L3]

2.3.4. Assessment of Software Assurance Processes and Practices [L2/3]

3. Assurance Assessment

3.1. Assurance Assessment Concepts

3.1.1. Baseline level of assurance; allowable tolerances, if quantitative [L1]

3.1.2. Assessment methods [L2/3]

3.2. Measurement for Assessing Assurance

3.2.1. Product and process measures by life-cycle phase [L1/2]

3.2.2. Other performance indicators that test for the baseline as defined in 3.1.1, by life-cycle phase [L1/2]

3.2.3. Measurement processes and frameworks [L2/3]

3.2.4. Business survivability and operational continuity [L2]

3.3. Assurance Assessment Process (collect and report measures that demonstrate the baseline as defined in 3.1.1.)

3.3.1. Comparison of selected measurements to the established baseline [L3]

3.3.2. Identification of out-of-tolerance variances [L3]

4. Assurance Management

4.1. Making the Business Case for Assurance

4.1.1. Valuation and cost/benefit models, cost and loss avoidance, return on investment [L3]

4.1.2. Risk analysis [L3]

4.1.3. Compliance justification [L3]

4.1.4. Business impact/needs analysis [L3]

4.2. Managing Assurance

4.2.1. Project management across the life cycle [L3]

4.2.2. Integration of other knowledge units [L2/3]

4.3. Compliance Considerations for Assurance

4.3.1. Laws and regulations [L3]

4.3.2. Standards [L3]

4.3.3. Policies [L2/3]

5. System Security Assurance

5.1. For Newly Developed and Acquired Software for Diverse Systems

5.1.1. Security and safety aspects of computer-intensive critical infrastructure [L2]

5.1.2. Potential attack methods [L3]

5.1.3. Analysis of threats to software [L3]

5.1.4. Methods of defense [L3]

5.2. For Diverse Operational (Existing) Systems

5.2.1. Historic and potential operational attack methods [L4]

5.2.2. Analysis of threats to operational environments [L3]

5.2.3. Designing of and plan for access control, privileges, and authentication [L3]

5.2.4. Security methods for physical and personnel environments [L4]

5.3. Ethics and Integrity in Creation, Acquisition, and Operation of Software Systems

5.3.1. Overview of ethics, code of ethics, and legal constraints [L4]

5.3.2. Computer attack case studies [L3]

6. System Functionality Assurance

6.1. Assurance Technology

6.1.1. Technology evaluation [L3]

6.1.2. Technology improvement [L3]

6.2. Assured Software Development

6.2.1. Development methods [L2/3]

6.2.2. Quality attributes [L3—depends on the property]

6.2.3. Maintenance methods [L3]

6.3. Assured Software Analytics

6.3.1. Systems analysis [L2 architectures; L3/4 networks, databases (identity management, access control)]

6.3.2. Structural analysis [L3]

6.3.3. Functional analysis [L2/3]

6.3.4. Analysis of methods and tools [L3]

6.3.5. Testing for assurance [L3]

6.3.6. Assurance evidence [L2]

6.4. Assurance in Acquisition

6.4.1. Assurance of acquired software [L2]

6.4.2. Assurance of software services [L3]

7. System Operational Assurance

7.1. Operational Procedures

7.1.1. Business objectives [L3]

7.1.2. Assurance procedures [L3]

7.1.3. Assurance training [L4]

7.2. Operational Monitoring

7.2.1. Monitoring technology [L4]

7.2.2. Operational evaluation [L4]

7.2.3. Operational maintenance [L3]

7.2.4. Malware analysis [L2/3]

7.3. System Control

7.3.1. Responses to adverse events [L3/4]

7.3.2. Business survivability [L3]

After we completed the mapping of the current CERT research projects to the MSwA2010 BoK and selected it as the initial BASF, our next task was to see whether we could successfully use the BASF to perform gap analysis.

8 Gap Analysis for Identification of Promising Research Areas

Once we mapped the current CERT research projects to the MSwA2010 BoK (see Section 6) we performed an initial gap analysis to identify some promising research areas for CERT. For those areas not represented at all in the current CERT research projects, we checked the maturity level and assessed whether this work was being covered elsewhere. For example, Software Life-Cycle Processes (BoK area 1.1) is fairly mature *and* is being addressed by the SEI's process management program, among others. Therefore, we did not identify it as a gap area for CERT research. In other cases, there has been a fair amount of research work, but more is needed; Assured Software Development (BoK area 6.2) is an example. With this in mind, here is our initial list of gap areas:

- 3.1.1 Baseline level of assurance; allowable tolerances, if quantitative [L1]

  This relates to the gap in measurement work discussed below.
- 3.2.1 Product and process measures by life-cycle phase [L1/2]
- 3.2.2 Other performance indicators that test for the baseline, by life-cycle phase [L1/2]

  We have started to do some measurement work, but more is needed.
- 4.1 Making the Business Case for Assurance

  Methods for making the business case exist, such as calculating cost/benefit, but the data to support it is lacking.
- 6.2 Assured Software Development

  We are doing some work in this area, but more is needed.
- 6.3 Assured Software Analytics

  We are doing some work in this area, but more is needed.
- 6.4 Assurance in Acquisition

  We are doing some work in this area, but more is needed.

There are some areas of research that do not fit the BASF neatly. The BASF is not intended to exclude these areas, but we recognize that some important research work does not fit the MSwA2010 topics directly. For example, our recent software assurance curriculum work is needed research, but it does not map directly to the MSwA2010 topics. As another example, some of our advanced work in intrusion detection and network analysis also does not map directly to these topics. This may suggest the need for follow-on work to broaden the BASF to provide a framework for a wider range of research activities.

9 Conclusion and Future Plans

We began developing a framework for building assured systems by first considering customer pain points. We examined a number of existing life-cycle process models, security models, and security research frameworks. We then proposed that the MSwA2010 BoK areas could provide the bulk of the BASF. In order to test this hypothesis, we assigned maturity levels to each of the knowledge areas. We then mapped the CERT research projects to the BoK areas. We used the results of this mapping to perform a gap analysis to identify areas where additional research would be needed. The benefit of this approach is that it establishes a desirable linkage between software assurance research and the associated educational curriculum research. Advances in specific software assurance research areas could suggest changes to the MSwA2010 BoK. In turn, new BoK areas in the curriculum could suggest fruitful paths for additional software assurance research.

The original customer pain points that we set out to address are

1. How do I decide which security methods fit into a specific life-cycle activity?
2. How do I know if a specific security method is sufficiently mature for me to use on my projects?
3. When should I take a chance on a security research approach that has not been widely used?

We also wanted to address a more generic problem, one that we had seen in our own work and elsewhere, that various research projects in building assured systems appear unrelated to one another; we and other research entities consequently do not have a good way to prioritize and select new research.

The BASF helps to address some, but not all, of the customer pain points. It is helpful in addressing the first and second questions, but is limited in its usefulness in addressing the third question. Since the BASF naturally covers the development life cycle, mapping a particular method to the appropriate knowledge area(s) will help to answer the first question. In this report we have provided such a mapping for our research projects. It should be relatively easy to perform such a mapping for methods under consideration for use. For the second question, using knowledge area maturity levels in conjunction with examining a specific method will provide information up front so that a user can decide whether the method is sufficiently mature. The third question is a bit harder to answer and requires more work on the part of the user. A cost/benefit analysis or risk assessment will help to answer the question of whether it is worth taking a chance on an approach that has not been widely used. Also, the user would have the benefit of looking at a range of approaches for a particular activity, and assessing whether a less mature approach provides significant benefit relative to a more mature approach.

From a research perspective, researchers could periodically consider rating the maturity levels of their methods. This would assist users in deciding which methods to use. It would also be helpful if researchers could collect and/or provide available cost/benefit data and encourage users to assist in such data collection. All too often users decide on a particular method but do not collect enough information to determine whether the benefit justified the cost. At the same time, the

smaller projects that researchers conduct on their own do not usually result in enough cost/benefit data to be sufficiently compelling.

We believe that the BASF provides an umbrella for CERT's research work in building assured systems and that it can be used to show how the various research efforts fit together. A future formal tradeoff analysis of the research roadmaps and frameworks studied in the literature with the current BASF would reinforce this. The BASF could then be extended to cover a broader research scope, providing a more natural fit for some of our advanced research work in intrusion detection and various types of analysis (e.g., network, protocol, data) as well as software assurance curriculum research. The gap analysis that we have done could be used to help select, and to some extent prioritize, new research. For example, if research is proposed for an area where there are a number of mature approaches, it would be helpful to understand why that research would be considered a good investment, compared to areas where there are no mature approaches. Since there is a lot of research aimed at building assured systems, we anticipate that this framework would need regular review and revision in order to stay current.

Appendix

Software Assurance Definitions

The following are software assurance definitions we reviewed while developing the BASF.

The following is the U.S. Department of Defense's (DoD) definition of systems assurance taken from "A DoD-Oriented Introduction to the NDIA's System Assurance Guidebook" [Popick 2010]:

> The justified measures of confidence that a 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. [1]
>
> [1] NDIA – Systems Assurance Committee. *Engineering for System Assurance*. Oct. 2008 www.acq.osd.mil/sse/docs/SA-Guidebook-v1-Oct2008.pdf

The following definition is taken from the NDIA conference paper "Engineering Improvement in Software Assurance: A Landscape Framework" [Brownsword 2009]:

> Environment of use
>
> Actual environment of use (not just the expected environment of use)
>
> Means evaluating robustness against unexpected use, threats, and changes in the environment"

The following is from the SEI webinar *Engineering Improvement in Software Assurance: A Landscape Framework* [Brownsword 2010]:

> Software assurance: a justified level of confidence that software-reliant systems function as intended within their operational environment

The following is the Committee on National Security Systems (CNSS) definition [CNSS 2010] Used in the DHS SwA website [DHS 2010b] and *Software Security Engineering* book [Allen 2008]:

> **Software assurance** (SwA) is the level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at any time during its life cycle, and that the software functions in the intended manner (from CNSS 4009 IA Glossary - see Wikipedia for definitions and descriptions).

The following is the SAFECode Software Assurance Definition [SAFECode 2008]:

> Confidence that software, hardware and services are free from intentional and unintentional vulnerabilities and that the software functions as intended.

The following excerpt is from the Software Security Assurance State-of-the-Art Report (SOAR) [Goertzel 2007].

2.1 Definition 1: Software Assurance

Until recently, the term *software assurance* was most commonly relating two software properties: *quality* (*i.e.*, "software assurance" as the short form of "software quality assurance"), and *reliability* (along with reliability's most stringent quality—safety). Only in the past 5 years or so has the term software assurance been adopted to express the idea of the assured security of software (comparable to the assured security of information that is expressed by the term "information assurance").

The discipline of software assurance can be defined in many ways. The most common definitions complement each other but differ slightly in terms of emphasis and approach to the problem of assuring the security of software.

In all cases, all definitions of software assurance convey the thought that software assurance must provide a reasonable level of *justifiable confidence* that the software will function correctly and predictably in a manner consistent with its documented requirements. Additionally, the function of software cannot be compromised either through direct attack or through sabotage by maliciously implanted code to be considered assured. Some definitions of software assurance characterize that assurance in terms of the software's trustworthiness or "high-confidence."

Several leading definitions of software assurance are discussed below.

Instead of choosing a single definition of software assurance for this report, we synthesized them into a definition that most closely reflects software security assurance as we wanted it to be understood in the context of this report—*Software security assurance: The basis for gaining justifiable confidence that software will consistently exhibit all properties required to ensure that the software, in operation, will continue to operate dependably despite the presence of sponsored (intentional) faults. In practical terms, such software must be able to resist most attacks, tolerate as many as possible of those attacks it cannot resist, and contain the damage and recover to a normal level of operation as soon as possible after any attacks it is unable to resist or tolerate.*

2.1.1 CNSS Definition

The ability to establish confidence in the *security* as well as the predictability of software is the focus of the Committee on National Security Systems (CNSS) definitions of software assurance in its National Information Assurance Glossary. [8] The glossary defines software assurance as—

> *The level of confidence that software is free from vulnerabilities, regardless of whether they are intentionally designed into the software or accidentally inserted later in its life cycle, and that the software functions in the intended manner.*

This understanding of software assurance is consistent with the use of the term in connection with information, *i.e.*, information assurance (IA). By adding the term software assurance to its IA glossary, CNSS has acknowledged that software is directly relevant to the ability to achieve information assurance.

The CNSS definition is purely descriptive: it describes what software must *be* to achieve the level of confidence at which its desired characteristics—lack of vulnerabilities and predictable execution—can be said to be assured. The definition does not attempt to prescribe the means by which that assurance can, should, or must be achieved.

2.1.2 DoD Definition

The Department of Defense's (DoD) Software Assurance Initiative's definition is identical in meaning to that of the CNSS, although more succinct—

> *The level of confidence that software functions as intended and is free of vulnerabilities, either intentionally or unintentionally designed or inserted as part of the software. [9]*

2.1.3 NASA Definition

The National Aeronautics and Space Administration (NASA) defines software assurance as—

> *The planned and systematic set of activities that ensure that software processes and products conform to requirements, standards, and procedures.*

The "planned and systematic set of activities" envisioned by NASA include—

- Requirements specification
- Testing
- Validation
- Reporting.

The application of these functions "during a software development life cycle is called software assurance." [10]

The NASA software assurance definition predates the CNSS definition but similarly reflects the primary concern of its community—in this case, safety. Unlike the CNSS definition, NASA's definition is both descriptive and prescriptive in its emphasis on the importance of a "planned and systematic set of activities." Furthermore, NASA's definition states that assurance must be achieved not only for the software itself but also the processes by which it is developed, operated, and maintained. To be assured, both software *and* processes must "conform to requirements, standards, and procedures."

2.1.3 DHS Definition

Like CNSS, the Department of Homeland Security (DHS) definition of software assurance emphasizes the properties that must be present in the software for it to be considered "assured," *i.e.*—

- Trustworthiness, which DHS defines, like CNSS, in terms of the absence of exploitable vulnerabilities whether maliciously or unintentionally inserted
- Predictable execution, which "provides justifiable confidence that the software, when executed, will function as intended. [11]

Like NASA, DHS's definition explicitly states that "a planned and systematic set of multidisciplinary activities" must be applied to ensure the conformance of both software and processes to "requirements, standards, and procedures." [12]

2.1.4 NIST Definition

The National Institute of Standards and Technology (NIST) defines software assurance in the same terms as NASA, whereas the required properties to be achieved are those included in the DHS definition: trustworthiness and predictable execution. NIST essentially fuses the NASA and DHS definitions into a single definition, thereby clarifying the cause-and-effect relationship between "the planned and systematic set of activities" and the expectation that such activities will achieve software that is trustworthy and predictable in its execution. [13]

2.2 Definition 2: Secure Software

DHS's *Security in the Software Life Cycle* defines secure software in terms that have attempted to incorporate concepts from all of the software assurance definitions discussed in Section 2.1 as well as reflect both narrow-focused and holistic views of what constitutes secure software. The document attempts to provide a "consensus" definition that has, in fact, been vetted across the software security assurance community [or at least that part that participates in meetings of the DHS Software Assurance Working Groups (WG) and DoD/DHS Software Assurance Forums]. According to *Security in the Software Life Cycle*—

> *Secure software cannot be intentionally subverted or forced to fail. It is, in short, software that remains correct and predictable in spite of intentional efforts to compromise that dependability.*

Security in the Software Life Cycle elaborates on this definition—

> *Secure software is designed, implemented, configured, and supported in ways that enable it to:*
>
> - Continue operating correctly in the presence of most attacks by either resisting the exploitation of faults or other weaknesses in the software by the attacker, or tolerating the errors and failures that result from such exploits
> - Isolate, contain, and limit the damage resulting from any failures caused by attack-triggered faults that the software was unable to resist or tolerate, and recover as quickly as possible from those failures.

The document then enumerates the different security properties that characterize secure software and clearly associates the means by which software has been developed with its security:

> *Secure software has been developed such that—*
>
> - *Exploitable faults and other weaknesses are avoided by well-intentioned developers.*
> - *The likelihood is greatly reduced or eliminated that malicious developers can intentionally implant exploitable faults and weaknesses or malicious logic into the software.*
> - *The software will be attack-resistant or attack-tolerant, and attack-resilient.*
> - *The interactions among components within the software-intensive system, and between the system and external entities, do not contain exploitable weaknesses.*

References

URLs are valid as of the publication date of this document.

[Allen 2008]
Allen, Julia H.; Barnum, Sean; Ellison, Robert J.; McGraw, Gary; & Mead, Nancy R. *Software Security Engineering: A Guide for Project Managers*. Addison-Wesley Professional, 2008.

[Babylon 2009]
Babylon, Ltd. *Definition of Framework*. http://dictionary.babylon.com/framework/ (2009).

[Bartol 2008]
Bartol, Nadya. *Practical Measurement Framework for Software Assurance and Information Security, Version 1.0*. Practical Software & Systems Measurement (PSM). http://www.psmsc.com/Prod_TechPapers.asp (2008).

[Brownsword 2009]
Brownsword, Lisa; Woody, Carol; Alberts, Christopher; & Moore, Andrew. "Achieving Acquisition Excellence Via Effective Systems Engineering."12th Annual Systems Engineering Conference, NDIA, October 26-29, 2009, San Diego, CA.

[Brownsword 2010]
Brownsword, Lisa & Woody, Carol. *Engineering Improvement in Software Assurance: A Landscape Framework* (SEI Webinar). Software Engineering Institute, Carnegie Mellon University, May 2010. http://www.sei.cmu.edu/library/abstracts/presentations/20100513webinar.cfm

[Caralli 2010]
Caralli, Richard A.; Allen, Julia H.; Curtis, Pamela D.; White, David W., & Young, Lisa R. *CERT® Resilience Management Model, Version 1.0: Resilient Technical Solution Engineering (RTSE).* Software Engineering Institute, Carnegie Mellon University, 2010. http://www.cert.org/resilience/rmm.html

[CERT 2010]
CERT. *2009 CERT Research Annual Report*. Software Engineering Institute, Carnegie Mellon University. http://www.cert.org/research/2009research-report.pdf (2010).

[CMMI Product Team 2006]
CMMI Product Team. *CMMI® for Development, Version 1.2* (CMU/SEI-2006-TR-008). Software Engineering Institute, Carnegie Mellon University, 2006. http://www.sei.cmu.edu/library/abstracts/reports/06tr008.cfm

[CMMI Product Team 2007]
CMMI Product Team. *CMMI® for Acquisition, Version 1.2* (CMU/SEI-2007-TR-017). Software Engineering Institute, Carnegie Mellon University, 2007. http://www.sei.cmu.edu/library/abstracts/reports/07tr017.cfm

[CNSS 2010]
Committee on National Security Systems (CNSS). *National Information Assurance (IA) Glossary: CNSS Instruction No. 4009*. http://www.cnss.gov/Assets/pdf/cnssi_4009.pdf (April 2010).

[Devanbu 2000]
Devanbu, Premkumar T. & Stubblebine, Stuart. "Software Engineering for Security: A Roadmap." *ICSE 2000, 22nd International Conference on Software Engineering, Future of Software Engineering Track.* Limerick, Ireland, June 2000.

[DHS 2008]
Department of Homeland Security (DHS) Software Assurance (SwA) Processes and Practices Working Group. *Process Reference Model for Assurance Mapping to CMMI-DEV V1.2*. https://buildsecurityin.us-cert.gov/swa/procwg.html (2008).

[DHS 2009]
Department of Homeland Security (DHS). *A Roadmap for Cybersecurity Research*. http://www.cyber.st.dhs.gov/docs/DHS-Cybersecurity-Roadmap.pdf (November 2009).

[DHS 2010a]
Department of Homeland Security (DHS) Software Assurance (SwA). *Measurement Working Group*. https://buildsecurityin.us-cert.gov/swa/measwg.html (2010).

[DHS 2010b]
Department of Homeland Security (DHS) Software Assurance (SwA). Software Assurance Community Resources and Information Clearinghouse. https://buildsecurityin.us-cert.gov/swa/ (2010).

[Ellison 2004]
Ellison, Robert J.; Moore, Andrew P.; Bass, Andrew P.; Klein, Mark H.; & Bachmann, Andrew P. *Security and Survivability Reasoning Frameworks and Architectural Design Tactics* (CMU/SEI-2004-TN-022). Software Engineering Institute, Carnegie Mellon University, 2004. http://www.sei.cmu.edu/library/abstracts/reports/04tn022.cfm

[ERCIM 2008]
European Research Consortium for Informatics and Mathematics (ERCIM) & European Commission. *Strategic Seminar: Engineering Secure Complex Software Systems and Services.* Brussels, Belgium, October 16, 2008.
http://www.ercim.eu/activity/strategic-seminar

[Goertzel 2007]
Goertzel, Karen Mercedes; Winograd, Theodore; McKinley, Holly Lynne; Oh, Lyndon; Colon, Michael; McGibbon, Thomas; Fedchak, Elaine; & Vienneau, Robert. *Software Security Assurance State-of-the-Art Report (SOAR).* Joint endeavor by IATAC (Information Assurance Technology Analysis Center) with DACS (Data and Analysis Center for Software). http://iac.dtic.mil/iatac/download/security.pdf (July 2007).

[Howard 2006]
Howard, Michael & Lipner, Steve. *The Security Development Lifecycle*. Microsoft Press, 2006.

[IPRC 2006]
International Process Research Consortium; Forrester, Eileen, ed. *A Process Research Framework.* Software Engineering Institute, Carnegie Mellon University, 2006.

[Jones 2009]
Jones, Nigel A. *Building In… Information Security, Privacy and Assurance—A High-Level Roadmap. Cyber Security Knowledge Transfer Network*. http://www.ktn.qinetiq-tim.net/content/files/events/2009-04-23_building-in-security-assurance-privacy.pdf (2009).

[Leiwo 1999]
Leiwo, Jussipekka. "Observations on Information Security Crisis" (Computer Science and Information Systems Reports Technical Reports TR-21). *Proceedings of the 22nd Information Systems Research Seminar in Scandinavia (IRIS22): "Enterprise Architectures for Virtual Organizations,"* Volume 2, ed. Kakola, Timo K., 313–324. Keuruu, Finland: August, 1999.

[Lipner 2005]
Lipner, Steve & Howard, Michael. *The Trustworthy Computing Security Development Lifecycle*. http://msdn.microsoft.com/en-us/library/ms995349.aspx (2005).

[Maughan 2010]
Maughan, Douglas. "The Need for a National Cybersecurity Research and Development Agenda." *Communications of the ACM 32*, 2 (January 2010). http://www.csl.sri.com/users/neumann/insiderisks08.html#220

[McGraw 2008]
McGraw, Gary & Chess, Brian. "Software [In]security: A Software Security Framework: Working Towards a Realistic Maturity Model." *InformIT*. (October 15, 2008): http://www.informit.com/articles/article.aspx?p=1271382

[McGraw 2010]
McGraw, Gary; Chess, Brian; & Migues, Sammy. *Building Security In Maturity Model BSIMM v2.0*. http://www.bsimm2.com/ (Accessed July 2010).

[Mead 2010]
Mead, Nancy R.; Allen, Julia H.; Ardis, Mark; Hilburn, Thomas B.; Kornecki, Andrew J.; Linger, Rick; & McDonald, James. *Software Assurance Curriculum Project Volume I: Master of Software Assurance Reference Curriculum* (CMU/SEI-2010-TR-005, ESC-TR-2010-005). Software Engineering Institute, Carnegie Mellon University, 2010. http://www.sei.cmu.edu/library/abstracts/reports/10tr005.cfm

[Microsoft 2010a]
Microsoft. *Microsoft Security Development Lifecycle*. http://www.microsoft.com/security/sdl/about/process.aspx (2010).

[Microsoft 2010b]
Microsoft. *Microsoft Security Development Lifecycle Version 5.0*. http://download.microsoft.com/download/F/2/0/F205C451-C59C-4DC7-8377-9535D0A208EC/Microsoft%20SDL_Version%205.0.docx (Updated March 31, 2010).

[NASCIO 2009]
National Association of State Chief Information Officers (NASCIO). *Desperately Seeking Security Frameworks – A Roadmap for State CIOs.* NASCIO, 2009. http://www.nascio.org/publications/documents/NASCIO-SecurityFrameworks.pdf

[Okubo 2007]
Okubo, Takao & Tanaka, Hidehiko. "Secure Software Development through Coding Conventions and Frameworks." *Second International Conference on Availability, Reliability and Security* (ARES'07). Fujitsu Laboratories Ltd., Institute of Information Security, 2007.

[O'Neil 2008]
Tsipenyuk O'Neil, Yekaterina; Chess, Brian; & West, Jacob. "JavaScript Hijacking: Only 1 Out 12 Popular AJAX Frameworks Prevents It." *AjaxWorld Magazine* (November 14, 2008). http://ajax.sys-con.com/node/747965

[OWASP 2009]
Open Web Application Security Project (OWASP). *Software Assurance Maturity Model (SAMM) v1.0*. http://www.owasp.org/index.php/Category:Software_Assurance_Maturity_Model (2009).

[Pavlich-Mariscal 2006]
Pavlich-Mariscal, J. A.; Demurjian, S. A.; & Michel, L. D. *A Framework for Composable Security Definition, Assurance, and Enforcement.* Springer Berlin/Heidelberg, 2006.

[Popick 2010]
Popick, Paul; Devine, Terence E.; & Moorthy, Rama. "A DoD-Oriented Introduction to the NDIA's System Assurance Guidebook." *CrossTalk 23*, 2 (Mar/April 2010). http://www.stsc.hill.af.mil/crosstalk/2010/03/1003PopickDevineMoorthy.html

[SAFECode 2008]
Software Assurance Forum for Excellence in Code (SAFECode). *Software Assurance: An Overview of Current Industry Best Practices.* http://www.safecode.org/publications/SAFECode_BestPractices0208.pdf (February 2008).

[SAFECode 2010]
Software Assurance Forum for Excellence in Code (SAFECode). *SAFECode*. http://www.safecode.org (2010).

[SEI 2010a]
Software Engineering Institute (SEI). *Capability Model Integration (CMMI).* Software Engineering Institute, Carnegie Mellon University. http://www.sei.cmu.edu/cmmi/ (2010).

[SEI 2010b]
Software Engineering Institute (SEI). *CMMI for Development (CMMI-DEV).* Software Engineering Institute, Carnegie Mellon University. http://www.sei.cmu.edu/cmmi/tools/dev/index.cfm (2010).

[SEI 2010c]
Software Engineering Institute (SEI). *CMMI for Acquisition*. Software Engineering Institute, Carnegie Mellon University. http://www.sei.cmu.edu/cmmi/tools/acq/index.cfm (2010).

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	September 2010	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Building Assured Systems Framework	FA8721-05-C-0003

6. AUTHOR(S)

Nancy R. Mead, Julia H. Allen

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2010-TR-025

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	ESC-TR-2010-025

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

Researchers at the CERT® Program, part of Carnegie Mellon University's Software Engineering Institute, need a framework to organize research and practice areas focused on building assured systems. The Building Assured Systems Framework (BASF) addresses the customer and researcher challenges of selecting security methods and research approaches for building assured systems. After reviewing existing life-cycle process models, security models, and security research frameworks, the authors used the Master of Software Assurance Reference Curriculum knowledge areas as the BASF. The authors mapped all major CERT research areas to the BASF, proving that the BASF is useful for organizing building assured systems research. The authors also performed a gap analysis to identify promising CERT research areas. The BASF is a useful structure for planning and communicating about CERT research. The BASF will also be useful to CERT sponsors to track current research and development efforts in building assured systems.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Assured Systems Development, Software Assurance, Framework	81

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL