Software Engineering Institute

Integrated Measurement and Analysis Framework for Software Security

Christopher Alberts
Julia Allen
Robert Stoddard

September 2010

TECHNICAL NOTE
CMU/SEI-2010-TN-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.

Table of Contents

Acknowledgments vii

Abstract ix

1 Introduction 1

2 Measurement Concepts 7
2.1 Measurement Process 8
2.1.1 Establish and Sustain Measurement Commitment 10
2.1.2 Plan for Measurement 10
2.1.3 Perform Measurement 11
2.1.4 Evaluate the Measurement Process 11
2.2 Using Information to Support Decision Making 11

3 Distributed Management Environments 14

4 Analyzing Performance in a Distributed Management Environment 17
4.1 System Decomposition and Event Analysis 17
4.2 Systemic Analysis 19

5 Mission-Objective-Driver (MOD) Analysis 21
5.1 Driver Identification 21
5.1.1 Mission 21
5.1.2 Objectives 21
5.1.3 Drivers 22
5.1.4 Deriving a Set of Drivers 23
5.1.5 A Standard Set of Drivers for Software Security 24
5.1.6 Tailoring an Existing Set of Drivers 25
5.2 Driver Analysis 26
5.3 Driver Profile 28

6 Integrated Measurement and Analysis Framework (IMAF) 31
6.1 Qualitative Implementation of the IMAF 32
6.2 Quantitative Implementation of the IMAF 33
6.3 Aligning Drivers with Software Security Codes of Practice 36

7 Insights and Next Steps 42
7.1 Summary 42
7.2 Surprises and Insights 42
7.3 Research Directions 44

Appendix A: Standard Set of Drivers for Software Security 45

Appendix B: Related Sources and Efforts 57

Glossary 59

References 61

List of Figures

Figure 1:	Measurement Process [ISO 2007]	9
Figure 2:	Measurement Must Focus on Effective Decision Making	12
Figure 3:	Acquisition Program	15
Figure 4:	Decision Making in Interactively Complex Socio-Technical Systems	18
Figure 5:	Decision Making Using Systemic Analysis	19
Figure 6:	Relationships Among Objectives and Drivers	24
Figure 7:	Driver Question and Range of Responses	26
Figure 8:	Driver Value Criteria	27
Figure 9:	Analyzed Driver	28
Figure 10:	Driver Profile: Programmatic Drivers	29
Figure 11:	Driver Profile: Product Drivers	30
Figure 12:	Integrated Measurement and Analysis Framework (IMAF)	31
Figure 13:	MOD Assessment	32
Figure 14:	Notional Bayesian Belief Network Showing Security Entity Relationships with a Security Objective	34
Figure 15:	Aligning Drivers with Software Security Codes of Practice	37

List of Tables

Table 1:	Example Software Security Measures by Life-Cycle Phase	4
Table 2:	Driver States	23
Table 3:	Prototype Set of Driver Questions for Software Security	25
Table 4:	Align Drivers with Software Security Codes of Practice	38
Table 5:	Part 1 Example for Driver 10 Security Requirements: Do Requirements Sufficiently Address Security?	40

Acknowledgments

The authors thank Archie Andrews, Carol Woody, and Dave Zubrow for their sponsorship and support of this work and for their review comments. We thank Michele Moss of Booz Allen Hamilton for her review comments that prompted the inclusion of Appendix B, and the chief scientist of the Software Engineering Institute's (SEI's) CERT® Program, Greg Shannon, for asking us the question that resulted in Section 7.2: "What has surprised you the most so far?" We also thank Audrey Dorofee for her technical contribution to the development of the MOD Analysis Method as part of the SEI's Mission Success in Complex Environments (MSCE) special project. Finally, the authors thank the SEI's CERT and Acquisition Support Programs for providing the funding to conduct this research effort.

Abstract

In today's business and operational environments, multiple organizations routinely work collaboratively to acquire, develop, deploy, and maintain technical capabilities via a set of interdependent, networked systems. Measurement in these distributed management environments can be an extremely challenging problem. The CERT® Program, part of Carnegie Mellon University's Software Engineering Institute (SEI), is developing the Integrated Measurement and Analysis Framework (IMAF) to enable effective measurement in distributed environments, including acquisition programs, supply chains, and systems of systems. The IMAF defines an approach that integrates subjective and objective data from multiple sources (targeted analysis, reports, and tactical measurement) and provides decision makers with a consolidated view of current conditions. This report is the first in a series that addresses how to measure software security in complex environments. It poses several research questions and hypotheses and presents a foundational set of measurement concepts. It also describes how meaningful measures provide the information that decision makers need when they need it and in the right form. Finally, this report provides a conceptual overview of the IMAF, describes methods for qualitatively and quantitatively collecting data to inform the framework, and suggests how to use the IMAF to derive meaningful measures for analyzing software security performance.

1 Introduction

Scientist Lord Kelvin said, "When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely in your thoughts advanced to the stage of science."[1] He also is quoted as having said, "When you cannot measure it, you cannot improve it."[2]

Many organizations measure just for the sake of measuring, with little or no thought given to what purpose and business objectives are being satisfied or what questions each measure is intended to inform. Meaningful measurement is about transforming strategic direction, policy, and other forms of management decision into action and measuring the performance of such action. According to the Corporate Information Security Working Group, "Visible measures provide a positive influence on human behavior by invoking the desire to succeed and compare favorably with one's peers" [CISWG 2005].

The right measures express the extent to which objectives are being met, how well requirements are being satisfied, how well processes and controls are functioning, and the extent to which performance outcomes are being achieved. The general purpose of measurement and analysis is to provide decision makers with the information they need when they need it and in the right form. For researchers, the purpose of measurement and analysis is to enable the testing of hypotheses while examining individual cause-effect or leading-indicator relationships. Such relationships eventually form a holistic model that can be used to predict the extent to which software security objectives can be achieved.

Since the mid-1990s, the CERT® Program at Carnegie Mellon University's Software Engineering Institute (SEI) has researched and created value-added processes, methods, practices, and tools for software survivability, software assurance, and building security into software throughout its development life cycle. In recent years, the research community has increasingly contributed to the body of knowledge about software assurance and software security measures and indicators.[3]

Unfortunately, the security community often assumes that information security includes software security, and thus that information security measures include software security measures. These disciplines are, in fact, quite distinct, as expressed in the following definitions from the Committee on National Security Systems' *National Information Assurance Glossary* [CNSS 2010],

> *Information security: The protection of information and information systems from unauthorized access, use, disclosure, disruption, modification, or destruction in order to provide confidentiality, integrity, and availability.*

---

1 http://en.wikiquote.org/wiki/William_Thomson

2 http://zapatopi.net/kelvin/quotes/

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

3 In this report, we use the terms *measures* and *indicators* rather than *metrics*. The ISO measurement community has discontinued use of the term *metrics* because it has become overloaded in use and meaning.

> *Software assurance: Level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at anytime during its lifecycle and that the software functions in the intended manner.*

and the following definition from *Software Assurance Curriculum Project Volume I: Master of Software Assurance Reference Curriculum* [Mead 2010]:

> *Software security: Engineering software so that it is as vulnerability- and defect-free as possible and continues to function correctly in spite of attack or misuse.*

Information security (and its measurement) focuses on protecting the confidentiality, availability, and integrity of information and information systems in operational production environments. By contrast, software security (and its measurement) focuses on a much earlier phase in the system development and system acquisition life cycles. While information security measures and software security measures inform and influence one another, they are not the same thing.

Efforts to identify and deploy meaningful information and operational security measures have been occurring for some time. These efforts include various reports by the U.S. National Institute of Standards and Technology [Chew 2008], the Workshop on the Economics of Information Security (WEIS) [WEIS 2010], and consensus efforts such as those conducted by the Center for Internet Security [CIS 2010] and the Corporate Information Security Working Group [CISWG 2005]. Corresponding efforts to identify and deploy meaningful software security measures have yet to materialize in any substantive fashion, although foundational definition work has been performed (refer to Appendix B). Significant work has been done to identify practices for developing more secure software (refer to Section 6.3), but work is still emerging that measures the extent to which such practices increase confidence or ensure that software will function more securely in its operational environment (as defined above).

Consequently, in Fiscal Year[4] 2010 (FY10), CERT began new research in software security measurement and analysis that builds on its core competence in software and information security as well as the SEI's Software and Engineering and Process Management Program work in software engineering measurement and analysis [SEI 2010]. The purpose of this research project is to address the following three questions:

- Q1: How do we establish, specify, and measure justified confidence that a software product[5] is sufficiently secure to meet operational needs?
- Q2: How do we measure at each phase of the development or acquisition life cycle whether the required/desired level of security has been achieved?
- Q3: How do we scale measurement and analysis approaches to complex environments, such as large, distributed systems of systems?

In essence, these questions ask how we move from uncertainty to justified confidence with respect to the security of software in its operational environment. Our intent is that these questions focus on ensuring that decision makers (development program and project managers, acquisition pro-

[4] The government fiscal year (and thus the SEI fiscal year) runs from October 1 of a given year to September 30 of the next year.

[5] For example, a software application, set of applications, software-reliant system, or system of systems.

gram offices) get the information they need when they need it and in the right form to determine where best to invest their time and resources.

Answering the first question will define the baseline against which software security can be measured. This will create a meaningful expression of the degree or level of software security for a specific set of related software components or systems. Ideally, this expression and its measurement should be generated as part of initial planning and specification, not as an afterthought during testing, integration, and deployment. In addition to specifying software security requirements, risk analysis approaches, including the prioritization of software components and systems based on their contribution to mission success, are also relevant. The SEI has produced promising results by applying methods for managing risks and opportunities,[6] as well as methods such as assurance cases[7] for capturing this expression. The current research project expands upon prior work by proposing a measurement and analysis framework that can be applied in single-organization or single-system environments, as well as interactively complex, socio-technical environments that span multiple organizational entities where management is distributed. While we apply the framework to software security in this report, we are discovering that it is also applicable to other domains of interest such as software supply chain and incident management.

Given a baseline against which to measure, approaches to the second question will include key product measures, process measures, and performance indicators that can be used to validate the required level of software security appropriate to a given life-cycle phase. Such measures will be developed within the context of a measurement process and framework that can be tailored for a specific development project. Table 1 presents hypothetical examples of life-cycle-phase measures that could aid in demonstrating required levels of software security. A method for selecting and deriving such measures is described in Section 6.3.

[6] The SEI's work in risk and opportunity management is described at http://www.sei.cmu.edu/risk/index.cfm.

[7] The SEI's work in assurance cases is described at http://www.sei.cmu.edu/dependability/tools/assurancecase/index.cfm.

Table 1: Example Software Security Measures by Life-Cycle Phase

Life-Cycle Phase	Example Software Security Measures
Requirements engineering	• Percentage of relevant software security principles reflected in requirements specifications (assuming security principles essential for a given development project have been selected) • Percentage of security requirements that have been subject to analysis (risk, feasibility, cost-benefit, performance tradeoffs) prior to being included in the specification • Percentage of security requirements covered by attack patterns, misuse/abuse cases, and other specified means of threat modeling and analysis
Architecture and design	• Percentage of architectural/design components subject to attack-surface analysis and measurement • Percentage of architectural/design components subject to architectural risk analysis • Percentage of high-value security controls covered by security design patterns
Coding	• Percentage of software components subject to static and dynamic code analysis against known vulnerabilities and weaknesses • Percentage of defects discovered during coding where the defects were injected in architecture and design or injected in requirements specification • Percentage of software components subject to code integrity and handling procedures, such as chain of custody verification, anti-tampering, and code signing
Testing	• Percentage of defects discovered during testing where the defects were injected in coding, in architecture and design, or in requirements specification • Percentage of software components with demonstrated satisfaction of security requirements as represented by a range of testing approaches (functional, risk-based, fuzz, penetration, black box, white box, code coverage, etc.) • Percentage of software components that demonstrated required levels of attack resistance and resilience when subject to attack patterns, misuse/abuse cases, and other specified means of threat modeling and analysis

The scope of the research project, for FY10 and beyond, is the three research questions stated above. In this report, and in FY10, we focus primarily on the first research question to establish a sound foundation. We have begun some exploratory work on the second question, which is described in Section 6.3. We have yet to commence work on question 3 other than accounting for it in the development and definition of the measurement analysis framework that has resulted from question 1.

Report Objective

In this report, we propose a new measurement and analysis framework that is focused on improving decision making related to software security. This framework integrates approaches that are being used successfully in a wide range of disciplines and offers new approaches that address the challenges that arise when considering the large-scale systems of systems and distributed management environments within which most of today's software-reliant systems operate.

This report represents the beginning of a dialogue on the research questions presented above. The results presented herein are not conclusive. Section 7, "Next Steps," summarizes activities in FY11 to pilot, analyze, and improve this work.

Intended Audience

This report is written for

- those engaged in research in software security measurement and analysis and related disciplines as they apply to meeting the information needs of decision makers

- decision makers who will use the results of this research (This includes managers of programs seeking to develop and acquire software and software-reliant systems and systems of systems.)

In addition, readers with the following types of experience may find the approaches presented in this report to be of interest:

- software engineering measurement and analysis
- program and project management
- organizational decision making
- risk management
- process improvement
- quality assurance

Report Content

Section 2, "Measurement Concepts," summarizes the background and foundations that this research builds upon. It describes a general process for measurement and analysis and identifies where this project's research questions fit within that process. This section introduces Figure 2, which describes the role of decision makers and how they currently obtain their information from multiple, often voluminous and conflicting, sources. We build upon this figure throughout the report as a foundation for the Integrated Measurement and Analysis Framework.

Section 3, "Distributed Management Environments," defines key terms used throughout this report (project, system of systems, distributed management environment) and sets the scope of this research project. The scope ranges from single-software, single-system, single-organization entities to multiple independently managed organizational entities working collaboratively to achieve a common mission or purpose using software-reliant systems and systems of systems.

Section 4, "Analyzing Performance in a Distributed Management Environment," describes how the scope and concepts identified in the previous section are applied. It compares and contrasts traditional analysis approaches that fall under the broad heading of system decomposition and event analysis with broader, more holistic approaches derived from system theory (systemic analysis). This section makes the case that an approach for measuring performance based on systemic analysis is better able to address the challenges that arise in distributed management environments. That said, both systemic and tactical methods for measuring performance are required.

Section 5, "Mission-Objective-Driver (MOD) Analysis," presents a method for performing systemic analysis of interactively complex socio-technical systems across the system development and system acquisition life cycles. The two activities that form the foundation of MOD analysis, driver identification and driver analysis, are defined and described. A starter set of drivers for software security are presented (with supporting details in Appendix A). This section promotes the use of drivers as the means for "connecting the dots" in tracing missions to objectives to drivers to measures and practices for software security.

Section 6, "Integrated Measurement and Analysis Framework (IMAF)," builds upon all prior sections, presenting the framework that serves as our current response to research question 1. The IMAF uses MOD Analysis to integrate subjective and objective data from multiple sources (tar-

geted analysis, reports, and tactical measurement) and provide decision makers with a consolidated view of current conditions. This section describes qualitative and quantitative implementations of the framework that will be used when piloting and validating it. It also defines a two-part approach to analyzing software security codes of practice against the starter set of software security drivers, with the objective of defining meaningful measures that will inform decision makers.

Section 7, "Insights and Next Steps," describes the research we intend to pursue in FY11. It also includes a few thoughts on what has surprised us as we have performed this work.

2 Measurement Concepts

The SEI has engaged in software engineering measurement and analysis for many years, and we drew from this body of knowledge to inform this project and report. *Goal-Driven Software Measurement—A Guidebook* [Park 1996] and the SEI's website [SEI 2010] state the following as the foundation for measurement and analysis:

> **Why measure?** *Because without data, you only have opinions.*
>
> **Why analyze?** *Because the data you collect can't help you if you don't understand it and use it to shape your decisions.*
>
> *Measurement and analysis involves gathering quantitative data about products, processes, and projects and analyzing that data to influence your actions and plans.*
>
> *Measurement and analysis activities allow you to*
>
> - **characterize**, *or gain understanding of your processes, products, resources, and environments and to establish baselines for comparisons with future assessments.*
> - **evaluate**, *to determine your status with respect to your plans. Measures are the sensors that let us know when our projects and processes are drifting off track, so that we can bring them back under control. We also evaluate to assess achievement of quality goals and to assess the impacts of technology and process improvements on products and processes.*
> - **predict**, *by understanding relationships among processes and products and building models of these relationships, so that the values we observe for some attributes can be used to predict others. We do this because we want to establish achievable goals for cost, schedule, and quality—so that appropriate resources can be applied. Predictive measures are also the basis for extrapolating trends, so estimates for cost, time, and quality can be updated based on current evidence. Projections and estimates based on historical data also help us analyze risks and make design/cost tradeoffs.*
> - **improve**, *by identifying roadblocks, root causes, inefficiencies, and other opportunities for improving product quality and process performance. Measures also help us plan and track improvement efforts. Measures of current performance give us baselines to compare against, so that we can judge whether or not our improvement actions are working as intended and what the side effects may be. Good measures also help us communicate goals and convey reasons for improving. This helps engage and focus the support of those who work within our processes to make them successful.*

Douglas Hubbard in his book *How to Measure Anything* [Hubbard 2007] defines measurement as "a set of observations that reduce uncertainty where the result is expressed as a quantity." Prior to measuring, he states that we should ask the following five questions [Hubbard 2007]:

1. What is the decision that the measurement is supposed to support?
2. What really is the thing being measured?
3. Why does this thing matter to the decision being made?
4. What do you know about it now?
5. What is the value to measuring it further?

He also states that "If a measurement matters at all, it is because it must have some conceivable effect on decisions and behavior. If we can't identify what decisions could be affected by a proposed measurement and how that measurement could change them, then the measurement simply has no value" [Hubbard 2007].

With these stated foundations, we describe a general measurement process and how such a process and the information it produces can be used to support decision making.

2.1 Measurement Process

A process for measurement and analysis (MA) defines, implements, and sustains a measurement capability, including its individual activities, that satisfies the information needs of decision makers within an organizational entity. For the purpose of this research project and report, an organizational entity may be of a size and complexity ranging from a single organization up to and including multiple, independently managed organizations that are working collaboratively to achieve a common mission such as a global supply chain (refer to Section 3).

An MA process typically involves the following [CMMI Product Team 2006]:

- Specify the objectives for measurement and analysis such that they are aligned with identified information needs and objectives.
- Specify the measures, analysis techniques, and mechanisms for data collection, data storage, data reporting, and feedback.
- Implement the collection, storage, analysis, and reporting of the data.
- Provide objective results that can be used in making informed decisions and take appropriate corrective actions.

The types of activities included in an MA process generally include [ISO 2007]:

- Establish and sustain commitment to the measurement program and process.
- Plan for measurement.
- Perform measurement.
- Evaluate the measurement process.
- Improve the measurement process and other processes that it informs (in satisfying information needs).

These activities and their relationships are shown in Figure 1, which is adapted from *ISO/IEC 15939:2007 Systems and Software Engineering – Measurement Process* [ISO 2007]. A version of this figure also appears in *Practical Software Measurement: Objective Information for Decision Maker*s [McGarry 2002].

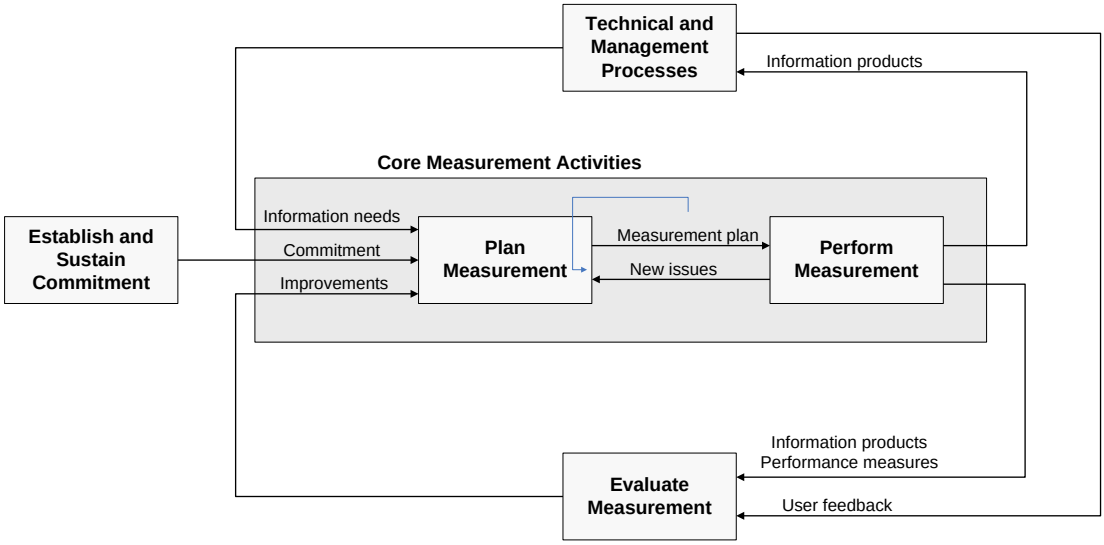

Figure 1: Measurement Process [ISO 2007]

Successful implementation of an MA process results in the following [ISO 2007]:

- Commitment for measurement is established and sustained across the organizational entity.
- The information needs of decision makers, and the technical and management processes that support them, are identified.
- An appropriate set of measures driven by the information needs are identified and/or developed.
- Measurement activities are identified.
- Identified measurement activities are planned.
- The required data is collected, stored, and analyzed, and the results are interpreted.
- Information products are used to support decisions and provide an objective basis for communication.
- The measurement process and measures are evaluated.
- Improvements are communicated to the measurement process owner.

The research results described in this first report do not address all MA process activities. The scope for this effort is the research questions described in Section 1; for FY10, we have focused on addressing research question 1, "How do we establish, specify, and measure justified confidence that a software product is sufficiently secure to meet operational needs?"

Research question 1 maps to *plan for measurement* process activities. Research question 2, "How do we measure at each phase of the development or acquisition life cycle that the required/desired level of security has been achieved?" maps to *perform measurement*, which will be an area of focus for FY11.

All MA process activities are briefly described here so that *plan for measurement* and *perform measurement* are presented in the context of the full process. As this work proceeds, all process activities will be addressed. Detailed descriptions of each of the process steps summarized here

are available in *ISO/IEC 15939: 2007* [ISO 2007] and *Practical Software Measurement* [McGarry 2002].

2.1.1 Establish and Sustain Measurement Commitment

No measurement process can succeed without management and stakeholder commitment, both up front as the process is being scoped and defined and on an ongoing basis as the process is implemented. That commitment requires a sponsor who ensures that decision makers and key stakeholders are fully engaged. The sponsor works with these individuals to allocate the resources necessary to execute all process activities on a sustaining basis, to use the measurement reports that result from the process, and to identify improvements that will make results most useful for informing key decisions.

Additionally, a grassroots commitment to establishing and sustaining measurement must exist in the sense that each individual in the organization feels free to provide accurate and timely data. To achieve such a grassroots commitment, organizations must recognize the psychology of measurement and address any institutional fear of measurement. Individuals and projects must view measurement as a positive and purposeful activity deserving of the utmost discipline and quality. Additional policies that may be warranted include sufficient data security and usage controls, sometimes including a measurement code of ethics to be signed by all managers, data custodians, and other users of the data repository.

2.1.2 Plan for Measurement

The *plan for measurement* process activity encompasses (1) the identification of information needs for decision makers and (2) the selection of appropriate measures to address those needs. Planning for measurement considers a project's goals, constraints, risks, and issues or problems. Information needs can be derived from societal, political, environmental, economic, business, organizational, regulatory, technological, product, and programmatic objectives.

For the purpose of this research project, the scope of information needs and the decisions they inform are intended to cover a wide range of contexts for the measurement and analysis of software security, including

- a single-software application, a set of applications, a software-reliant system, and a system of systems
- software and systems that are being developed or acquired
- software and systems in operation including the modification of existing systems and the addition of new software and systems
- single organizations and multiple organizations collaborating to achieve a joint mission

Given the range of objectives and these broad, multidimensional contexts, Section 6 proposes a process and framework for identifying the information needs of decision makers and provides a few examples of how measures could be selected and derived to address information needs.

Planning for measurement also addresses the tasks, schedule, and resources (staff, technologies, facilities, etc.) required to accomplish all measurement process activities. This includes defining the procedures that will be used for data collection, storage, analysis, and reporting.

2.1.3 Perform Measurement

The *perform measurement* process activity encompasses the timely collection, analysis, storage, and reporting of measurement data to provide decision makers with the information products that satisfy their information needs. This activity is the heart of MA process implementation. Analysis and reporting includes formulating recommendations for decision makers and providing alternative courses of action based on measurement results.

2.1.4 Evaluate the Measurement Process

The *evaluate the measurement* process assesses both the measures that are used, as well as the capability of the measurement process itself. Evaluating a measurement process ultimately leads to the identification of improvements to the measurement effort. The measurement process may be evaluated in four ways as outlined in *Measurement and Analysis Infrastructure Diagnostic (MAID) Evaluation Criteria, Version 1.0* [SEMA 2009]:

1. measurement and analysis planning—an evaluation of the planning for measurement at various levels of the organization down to and including the project level
2. data collection and storage—an evaluation of the processes, responsibilities, and tools used to collect and store data
3. data analysis—an analysis of how an organization conducts data analysis including analytical methods and tools
4. measurement and analysis reporting—an evaluation of the processes, integrity, and effectiveness of reporting the results of measurement and analysis

When analyzing data using quantitative methods (discussed in Section 6.2), the quality of the data is particularly important. Initially, sample data will be used due to the lack of sufficient security-related data, so the quality of the sample data is essential to any subsequent hypothesis testing and predictive analytics. Too much noise in sample data tends to either mask significant relationships or project false relationships.

Improving the measurement process involves a wide variety of solutions based on identified deficiencies. Improvements can run the gamut from building proper senior management commitment and support for measurement to increasing the quality of collected measurement data. Common process aids used by teams in identifying measurement process improvements include the Ishikawa diagram[8] (otherwise known as the fishbone diagram) and Failure Modes and Effects Analysis (FMEA) [Stamatis 2003]. Both of these techniques structure the discussion about what can go wrong and why.

2.2 Using Information to Support Decision Making

A decision maker is the individual or management team that oversees a software development or acquisition program. The decision maker is accountable for ensuring that the end product or system meets its requirements within negotiated cost and schedule constraints and ensures that the product functions as intended in its operational environment, including issues with respect to security. For this research project, this means providing justified confidence that the software prod-

[8] http://en.wikipedia.org/wiki/Ishikawa_diagram

uct is sufficiently secure to meet operational needs once it is deployed. Information needs comprise the data and knowledge necessary to provide decision makers with the facts and insights to make informed decisions. Decision makers responsible for systems in operation are often not the same people as those responsible for development and acquisition, which can further complicate the identification of information needs.

Decision makers consume information from a wide variety of sources to inform their decisions. Unfortunately, in the internet age, information consumers can easily become overwhelmed. One of the key themes throughout this report is the determination of how to define and provide just the information that decision makers need when they need it and in the right form. The attention span of decision makers is a constrained resource, so we need approaches for directing their attention to the information that matters most.

In general, decision makers obtain information from multiple sources, as shown in Figure 2.

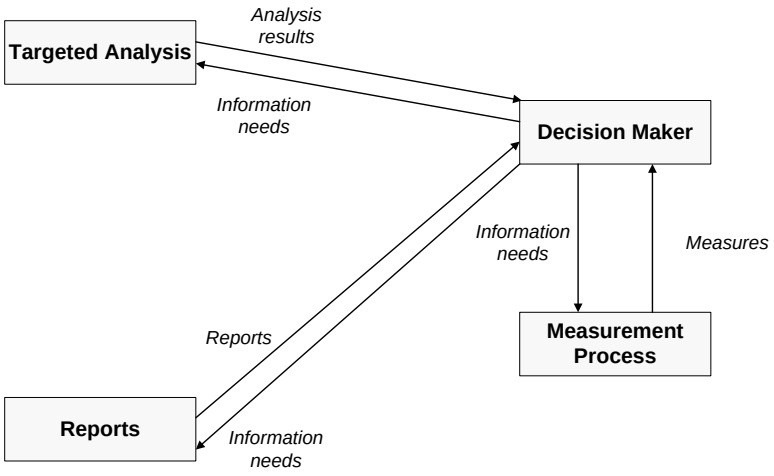

Figure 2: Measurement Must Focus on Effective Decision Making

The measurement process illustrated in Figure 2 is as described in Section 2.1. It defines activities that gather and analyze measurement data based on defined information needs. If obtained properly, measurement data (measures and indicators) provides the needed information to support effective decisions. The use of the measurement process to inform decision makers will evolve as this research project addresses each measurement process activity (as described in Section 2.1).

The box labeled "Targeted Analysis" includes information and knowledge that results from the application of analysis methods, techniques, and tools. Targeted analysis is conducted to support decision making based on defined information needs. This can be done independently and within the context of the measurement process.

Reports include textual and graphical information products that support defined information needs and are produced in the form and language that are meaningful for decision makers.

It is worth noting that measurement is a well-defined discipline and is being applied successfully to software development [SEI 2010] and software security and assurance (refer to Appendix B). However, the focus of measurement has traditionally been in the context of a single organizational entity and a single project. The next section will describe why research on software security mea-

surement and analysis needs to reflect today's realities, which include multiple organizations collaborating in a distributed management environment.

3 Distributed Management Environments

A logical starting point when looking at software security measurement is to consider it within the context of a project. In this report, a *project* is defined as a planned set of interrelated tasks that are executed over a fixed period of time and within certain constraints, such as cost or funding limitations. Unlike operations, which are repetitive and permanent (or semipermanent) functional tasks intended to produce products or provide services, projects are temporary in nature. A *software project* is an endeavor that is intended to produce a software product, such as a software application or software-reliant system, within a fixed period of time and within specified budget constraints. Typically, a single manager has final decision-making responsibility and authority for a project. As a result, the project manager is considered to be the key decision maker for the project.

In recent years, the nature of software products has evolved with the emergence of computer networks. The focus has shifted from producing stand-alone software products to providing technical capabilities within a larger system-of-systems context. Here, a *system of systems* is defined as a set or arrangement of interdependent systems that are related or connected (i.e., networked) to provide a given capability [Levine 2003].[9] The following characteristics are used to differentiate a system of systems from a very large, complex monolithic system [Maier 1996]:

- managerial independence—The management of each system within a system of systems is independent from the management of the other systems.
- operational independence—Each system within a system of systems provides useful functionality apart from other systems.
- evolutionary character—Each system within a system of systems grows and changes independently of other systems over time.
- emergent behavior—Certain behaviors of a system of systems arise from the interactions among the individual systems and are not embodied in any of the individual systems.
- geographic distribution—Individual systems within a system of systems are dispersed over large geographic areas.

Providing technical capabilities via a set of interdependent, networked systems (i.e., a system of systems) requires the coordinated efforts of multiple organizations. Rather than focusing on how a project team provides a stand-alone software product for a customer, we must shift our focus to how multiple project teams working collaboratively provide technical capabilities via a set of networked software products. An *acquisition program* is defined as a collection of individual projects that work collaboratively to provide technical capabilities via a set of networked software products (i.e., a system of systems).

Figure 3 depicts an acquisition program. The mission of an acquisition program is to acquire, develop, and deploy networked software products that function as intended and that are reliable,

[9] Software within a system of systems comes from a variety of sources, including open source, commercial-off-the-shelf (COTS), custom developed, and so on. An inadequate understanding of the pedigree of software components and problems that arise from integrating software components from multiple sources leads to risk and uncertainty in systems of systems.

safe, and secure. Figure 3 shows how project teams from multiple organizations work collaboratively to achieve that mission. In some cases, relationships among organizations are formally defined. For example, an acquirer can execute a formal contract with a supplier that governs the relationship between the two organizations. Typically, the acquirer provides a set of requirements, and the supplier develops a software product that meets those requirements. Another example of a formal agreement between organizations is when an acquirer licenses commercial-off-the-shelf (COTS) software from a supplier. A license outlines any terms and conditions regarding the acquirer's use of the software product. However, the supplier is free to update or make changes to the software as it sees fit, without considering the impact on its customers (i.e., its licensees).

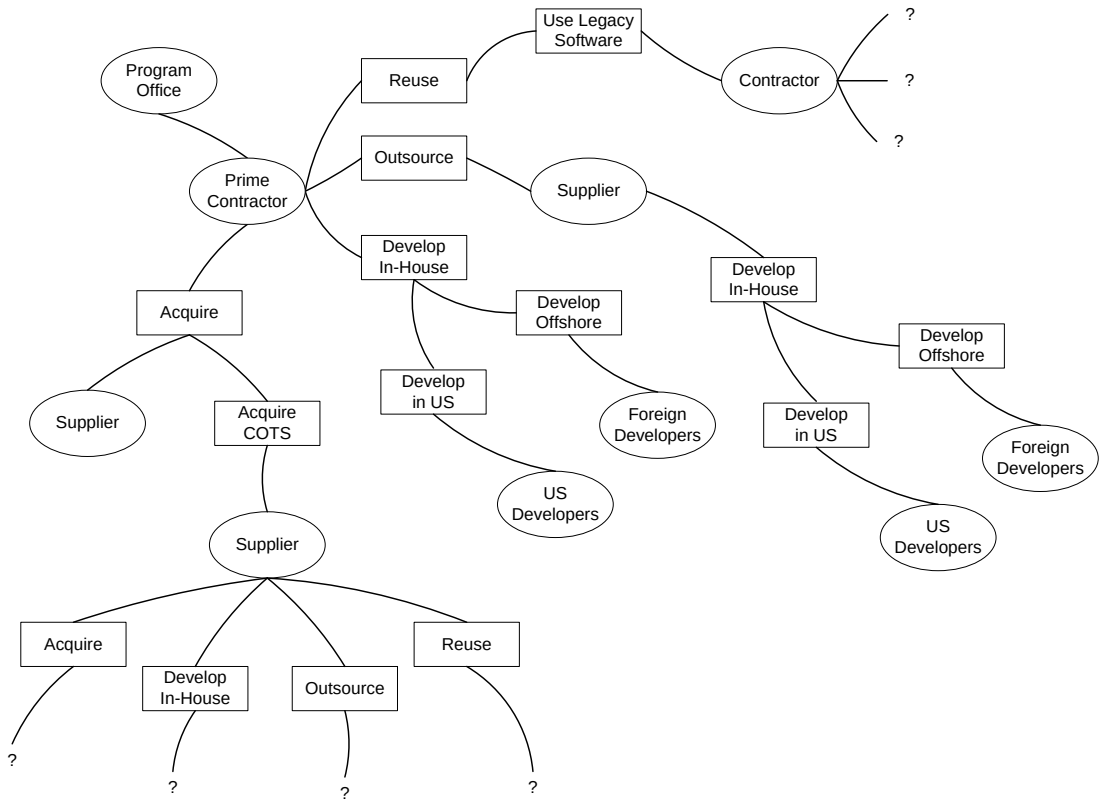

Figure 3: Acquisition Program

While many relationships within an acquisition program are governed by formal agreements, such as contracts or licenses, some relationships are informal. For example, software products being acquired and developed by a program are often required to interoperate with existing operational systems and with applications that are being acquired by other programs. In practice, relationships with other, separately funded programs tend to be informal and ad hoc.

An acquisition program is an example of a *distributed management environment*, which is defined as multiple, independently managed organizational entities working collaboratively to achieve a common mission or purpose. In general, no single administrative structure or set of policies governs all organizations in a distributed management environment such as an acquisition program. In addition, no single manager has authority over all organizations within the environment. Mul-

tiple points of management control (i.e., multiple decision makers) exist, which creates a degree of programmatic complexity that can be difficult to manage effectively.

The measurement approach outlined in this report can be used in all types of distributed management environments, including acquisition programs, supply chains, and systems of systems. However, our primary focus is on acquisition programs because of our interest in measuring software security early in the acquisition and development life cycles. The next section explores an approach for sorting through the programmatic and product complexity and measuring performance in distributed management environments.

4 Analyzing Performance in a Distributed Management Environment

Measurement and analysis should be tailored to the context in which it will be applied. Our research is focused on using measurement and analysis to assess performance in distributed management environments, such as an acquisition program.[10] The goal is to define an approach that is well suited to these environments by

- understanding the key characteristics of distributed management environments
- exploring alternative measurement and analysis approaches
- selecting an approach that is appropriate for distributed management environments

Establishing the key characteristics of a distributed management environment requires examining its core elements. A distributed management environment is an example of an interactively complex socio-technical system that spans multiple organizational entities. Here, a *socio-technical system* is defined as interrelated technical and social elements that are engaged in goal-oriented behavior. Elements of a socio-technical system include the people who are organized in teams or departments to do their work tasks and the technical systems on which people rely when performing work tasks.

Two distinct types of analysis can be used when evaluating interactively complex socio-technical systems [Leveson 2004]: (1) system decomposition and event analysis and (2) systemic analysis.

4.1 System Decomposition and Event Analysis

Most engineering analyses are based on the principle of system decomposition and event analysis [Leveson 2004]. The first step when conducting this type of analysis is to decompose the socio-technical system into its components. Individual components are then prioritized, and a subset of components is designated as being critical. Next, analysts evaluate how a series of predefined events might affect each critical component.

System decomposition and event analysis enables stakeholders to implement effective controls in order to mitigate potential failures triggered by a range of events. This approach is very useful when mitigating risks to critical components and, as a result, minimizing the likelihood that those components will fail. Overall, system decomposition and event analysis has proven to be an essential approach within the discipline of systems engineering. However, when using this approach to evaluate interactively complex socio-technical systems, analysts need to understand its limitations, which include the following [Leveson 2004]:

- The selection of which events to include in the analysis is subjective.

[10] An acquisition program is classified as a distributed management environment because it comprises multiple, independently managed organizational entities (i.e., projects) working collaboratively to achieve a common mission. However, a project is not an instance of a distributed management environment because, by definition, it is a single organizational entity and a distributed management environment requires the existence of multiple, independently managed organizational entities.

- An event's causal relationships are simple, direct, and linear. Nonlinear relationships, such as feedback, are not analyzed. In addition, only the proximate causes of failure are considered, and interactions among system components are not analyzed.
- Events that produce extreme or catastrophic consequences are difficult to predict because they can be triggered by the contemporaneous occurrences of multiple events, cascading consequences, and emergent system behaviors.

System decomposition and event analysis is focused on preventing the failure of critical components within a socio-technical system rather than on assuring the behavior of the system as a whole. As a result, it is not sufficient for evaluating the assurance of interactively complex socio-technical systems, like distributed management environments.

A relatively simple socio-technical system, as defined in this document, is one that has a small number of unknowns when considering interactions among system components and with the system's environment. System decomposition and event analysis is effective when applied to socio-technical systems that have few unknowns because uncertainty is relatively low and the range of events that can lead to failure is known. However, system decomposition and event analysis does not effectively handle the high degree of uncertainty inherent in interactively complex socio-technical systems. In addition, the degree of uncertainty in a socio-technical system tends to increase when it spans multiple organizations. Figure 4 shows that decision makers can have trouble making sense of numerous, disparate data when assessing assurance for interactively complex socio-technical systems.

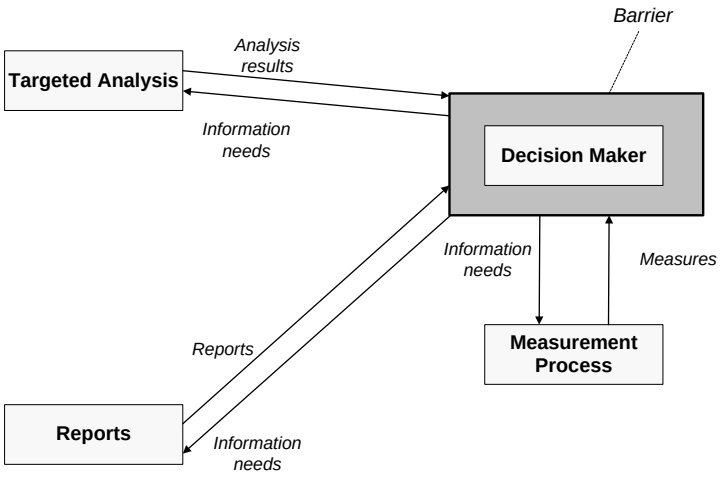

Figure 4: Decision Making in Interactively Complex Socio-Technical Systems

As illustrated in the figure, each source of information provides an insight into a piece of the overall system; however, the high degree of interactive complexity prevents the decision maker from "connecting the dots" among the data. As a result, the decision maker is unable to confidently assess the behavior of the system as a whole in terms of its ability to achieve mission success. A different approach is needed.

4.2 Systemic Analysis

Systemic analysis of socio-technical systems is based on system theory. The underlying principle of system theory is that a system is analyzed as a whole rather than decomposing it into individual components and then analyzing each component separately [Leveson 2004]. In fact, some properties of a system are best analyzed by considering the entire system, including

- influences of environmental factors
- feedback and nonlinearity among causal factors
- systemic causes of failure (as opposed to proximate causes)
- emergent properties

Systemic analysis thus assumes a holistic view of risk to an interactively complex socio-technical system. The first step in this type of analysis is to establish the objectives that must be achieved. The objectives define the desired outcome, or "picture of success," for a socio-technical system. Next, systemic factors that have a strong influence on the outcome (i.e., whether or not the objectives will be achieved) are identified. These factors, called *drivers* in this report, are important because they define a small set of factors that can be used to assess a socio-technical system's performance and gauge whether it is on track to achieve its key objectives. The drivers are then analyzed, which enables decision makers to gauge the performance (i.e., understand the behavior) of the system.

The SEI's experience shows that systemic analysis is useful for understanding the behavior of interactively complex socio-technical systems because it effectively handles their inherently high degree of uncertainty [Alberts 2009]. Applying systemic analysis to highly complex systems provides the decision maker with a means of confidently assessing the behavior of the system as a whole, which is necessary when assessing assurance. This concept is depicted in Figure 5 below, where systemic analysis enables decision makers to confidently assess the behavior of the system by integrating information obtained from multiple sources.

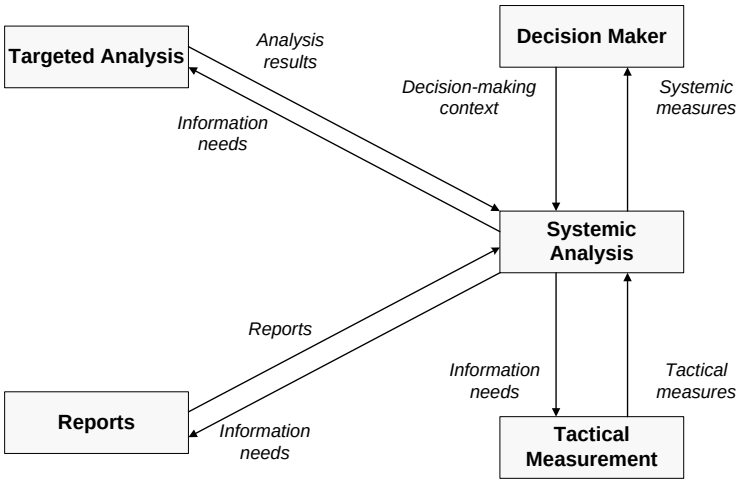

Figure 5: Decision Making Using Systemic Analysis

It is important to note that systemic analysis does not replace other types of measurement and analysis. Instead, it gathers data from many sources and then provides a consolidated view of per-

formance for decision makers. As a result, Figure 5 comprises a two-tiered measurement and analysis approach that produces two distinct types of measures: systemic and tactical. The diagram in Figure 5 builds on similar figures presented earlier in this report (Figure 2 and Figure 4) by incorporating systemic analysis into the measurement and analysis approach. Both types of measures will be briefly discussed, beginning with systemic measures.

In the scenario depicted in Figure 5, the decision maker has a thorough understanding of the problem space within which decisions must be made, which is referred to as the *decision-making context*. However, the decision maker is unable to directly articulate his or her information needs. *Systemic analysis* produces a set of drivers that can be used to assess a socio-technical system's performance and gauge whether it is on track to achieve its key objectives. The decision maker can then identify information needs based on the drivers that have been selected. Data from targeted analysis, reports, and tactical measurement that meet these information needs can be collected and analyzed. Evaluated drivers produce a set of systemic measures that provide the decision maker with insight into the overall performance of a socio-technical system. (An approach for evaluating drivers to produce systemic measures is presented in Section 5.2.)

The second key aspect of measurement and analysis depicted in Figure 5 is called *tactical measurement*. It defines an approach for producing a set of *tactical measures* that provide the decision maker with insight into a specific task that must be performed or into some characteristic of a work product. While tactical measurement provides useful information to decision makers about specific aspects of a socio-technical system, it does not provide direct insight into the system's overall potential for success. Systemic analysis integrates data from targeted analysis, reports, and tactical measurement and provides decision makers with relevant insights into the performance of the socio-technical system. (An approach for aligning tactical measures with drivers is presented in Section 6.3.) Because the framework provides relevant insights into the behavior of interactively complex socio-technical systems, the SEI is using it as a foundation for measuring performance in distributed management environments.

5 Mission-Objective-Driver (MOD) Analysis[11]

The SEI is developing the Mission-Objective-Driver (MOD) Analysis method to enable systemic analysis of interactively complex socio-technical systems. During preliminary development and testing activities, SEI researchers have determined that MOD Analysis enables an efficient and effective means of measuring performance in distributed management environments, such as acquisition programs [Alberts 2009, Dorofee 2008]. The following two activities form the foundation of MOD Analysis: (1) driver identification and (2) driver analysis. This section describes both activities in detail and then concludes with a discussion of the driver profile, which is the main output of MOD Analysis.

5.1 Driver Identification

The main goal of driver identification is to identify a set of factors, called drivers, that can be used to measure performance in relation to a program's mission and objectives. Once the set of drivers is identified, analysts can then evaluate each driver in the set to gain insight into the likelihood of achieving the mission and objectives. To measure performance effectively, analysts must ensure that the set of drivers conveys sufficient information about the mission and objectives being evaluated. As a result, the first step in identifying a set of drivers is to establish the mission.

5.1.1 Mission

MOD Analysis defines the term *mission* as the fundamental purpose of an individual, group, or operation. In the context of an acquisition program, the mission can be expressed in terms of the software product that is being acquired, developed, and deployed. The following is an example of a mission statement as required by MOD Analysis: *The XYZ Program is providing a new, web-based payroll system for our organization.*

The mission statement is important because it defines the target, or focus, of the measurement and analysis effort. After the basic target has been established, the next step is to identify which specific aspects of the mission need to be analyzed in detail.

5.1.2 Objectives

In MOD Analysis, an *objective* is defined as a tangible outcome or result that must be achieved when pursuing a mission. Each mission typically comprises multiple objectives. The goal of the second step of driver identification is to determine which of those objectives will be assessed during MOD Analysis. Selecting objectives refines the scope of the assessment to address specific aspects of the mission that are important to decision makers. In general, objectives identified during MOD Analysis should meet the following criteria:

- specific—The objective is concrete, detailed, focused, and well defined. It emphasizes action and states a specific outcome to be accomplished.
- measurable—The objective can be measured, and the measurement source is identified.

[11] Much of the material in this section is adapted from *A Framework for Categorizing Key Drivers of Risk* [Alberts 2009].

- achievable—The expectation of what will be accomplished is attainable given the time period, resources available, and so on.
- relevant—The outcome or result embodied in the objective supports the broader mission being pursued.
- time-bound—The timeframe in which the objective will be achieved is specified.

During driver identification, analysts must select one or more objectives that will be analyzed. The number of objectives depends on the breadth and nature of the issues being investigated. The following is an example of a generic objective for determining whether an acquisition program is adequately addressing software security: *When the system is deployed, security risks to the deployed system will be within an acceptable tolerance*. This example is fairly abstract; additional details must be added to the objective to meet the criteria listed above. For example, the objective could be augmented to address

- which system is being deployed
- when that system is expected to be deployed
- how risk will be measured
- how "acceptable tolerance" is defined for the program

The SEI's field experience shows that many decision makers (e.g., acquisition program managers) have difficulty constructing objectives that meet the above criteria for objectives. While decision makers have a tacit understanding of their objectives, they often cannot precisely articulate or express the objectives in a way that addresses the criteria. If the program's objectives are not clearly articulated, decision makers can have trouble assessing whether the program is on track for success. To address this issue, qualitative implementations of MOD Analysis allow for imprecise expressions of objectives. Specific information about objectives that is tacitly understood by program managers and staff becomes more explicit during execution of the MOD Analysis method. The remainder of Section 5 describes the qualitative implementation of MOD Analysis. A quantitative implementation of MOD Analysis is discussed in Section 6.2.[12]

5.1.3 Drivers

MOD Analysis defines a *driver* as a factor that has a strong influence on the eventual outcome or result (i.e., whether or not objectives will be achieved). Table 2 highlights three key attributes of a driver: *name*, *success state*, and *failure state*. The example driver in the table is named *Security Process*, and it examines how the program's processes are affecting achievement of the software security objective. Table 2 also indicates that each driver has two possible states: a success state and a failure state. The success state means that the program's processes incorporate security considerations adequately, which helps enable the achievement of the objectives. In contrast, the failure state signifies that the program's processes *do not* adequately incorporate security considerations and, as a result, the objectives will not be achieved.

[12] At this point in time, we do not have a good understanding of the relative values of using qualitative and quantitative implementations of MOD Analysis. A goal of our research is to provide guidance about the benefits of using each implementation.

Table 2: Driver States

Attribute	Description	Example
Name	A concise label that describes the basic nature of the driver.	Security process
Success state	A driver exerts a positive influence on the outcome.	The process being used to develop and deploy the system sufficiently incorporates security.
Failure state	A driver exerts a negative influence on the outcome.	The process being used to develop and deploy the system does not sufficiently incorporate security.

Analysis of a driver requires determining how it is currently acting (i.e., its current state) by examining the effects of conditions and potential events on that driver. The goal is to determine if the driver is

- almost certainly in its success state
- most likely in its success state
- equally likely to be in its success or failure states
- most likely in its failure state
- almost certainly in its failure state

This list defines the scale for driver analysis results. Analyzing each driver that contributes to a specific set of objectives establishes a benchmark of performance in relation to mission and objectives.

5.1.4 Deriving a Set of Drivers

The starting point for identifying a set of drivers is to articulate the mission and objectives that are being assessed. Analysts can then derive a set of drivers from them. The relationships among mission, objectives, and drivers are depicted in Figure 6.

Deriving a unique set of drivers based on the program's mission and objectives requires gathering information from people with experience and expertise relevant to the specified mission and objectives. For example, identifying a set of drivers for software development objectives requires input from acquisition programs managers and software-reliant systems developers. Similarly, analysts seeking to identify a set of drivers for software security would consult with security experts.

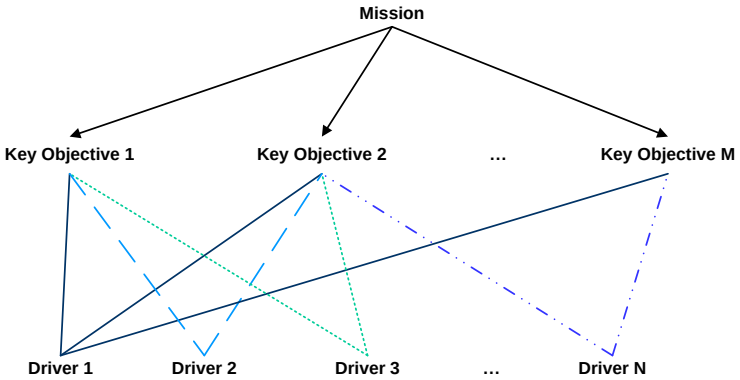

Figure 6: Relationships Among Objectives and Drivers

The experts from whom information is elicited should be familiar with the objectives that have been defined. Analysts can use the objectives to focus interviews or discussions with experts. During interviews or discussions, experts answer the following questions:

- What circumstances, conditions, and events will drive your program toward a *successful* outcome?
- What circumstances, conditions, and events will drive your program toward a *failed* outcome?

After they obtain information from the experts, analysts organize the information into approximately 10–20 groups that share the driver as the central idea or theme of each group. SEI staff has employed this approach for identifying drivers in a variety of areas, including software acquisition and development programs, cyber security processes, and business portfolio management [Alberts 2009]. The most recent focus has been on establishing drivers for software security. The next section presents a set of software security drivers that can be used as a starting point for tailoring measurement and analysis activities.

5.1.5 A Standard Set of Drivers for Software Security

The SEI has applied driver identification to software security. As a result, a standard set of 17 drivers for software security has been identified and documented. Table 3 lists the name of each software security driver along with a question that is used when analyzing that driver's state. These standard drivers were derived from the software security objective highlighted in Section 5.1.2 and have not been validated in pilot assessments. The next step in the development of the software security drivers is to validate them through field testing. Once a set of drivers is validated, it serves as an archetype that analysts can quickly tailor and apply to specific programs.

Table 3: Prototype Set of Driver Questions for Software Security

Driver Name		Driver Question
1.	Program Security Objectives	Are the program's security objectives realistic and achievable?
2.	Security Plan	Does the plan for developing and deploying the system sufficiently address security?
3.	Contracts	Do contract mechanisms with partners, collaborators, subcontractors, and suppliers sufficiently address security?
4.	Security Process	Does the process being used to develop and deploy the system sufficiently incorporate security?
5.	Security Task Execution	Are security-related tasks and activities performed effectively and efficiently?
6.	Security Coordination	Are security activities within the program coordinated appropriately?
7.	External Interfaces	Do work products from partners, collaborators, subcontractors, or suppliers meet security requirements?
8.	Organizational and External Conditions	Are organizational and external conditions facilitating completion of security tasks and activities?
9.	Event Management	Is the program able to identify and manage potential events and changing circumstances that affect its ability to meet its software security objectives?
10.	Security Requirements	Do requirements sufficiently address security?
11.	Security Architecture and Design	Do the architecture and design sufficiently address security?
12.	Code Security	Is the code sufficiently secure?
13.	Operational System Security	Is the integrated system sufficiently secure to support operations?
14.	Adoption Barriers	Have barriers to customer/user adoption of the system's security features been managed appropriately?
15.	Operational Security Compliance	Will the system comply with applicable security policies, laws, and regulations?
16.	Operational Security Preparedness	Are people prepared to maintain the system's security over time?
17.	Security Risk Tolerance	Is the security risk of the deployed system within an acceptable tolerance?

The drivers in Table 3 can be divided into two fundamental types: programmatic drivers and product drivers. Drivers 1–9 are referred to as *programmatic* drivers because they provide insight into how well the acquisition program is being managed. Drivers 10-17 are referred to as *product* drivers because they provide insight into the system that is being acquired, developed, and deployed.

5.1.6 Tailoring an Existing Set of Drivers

The standard set of drivers (Table 3) can be used to assess software security in general. However, the standard set must be to tailored to the requirements of a specific acquisition program to ensure that the

- set of drivers accurately reflects the key objectives of the specific program being assessed
- set of drivers is adjusted appropriately based on the program's context and characteristics
- phrasing of each driver is consistent with the program's terminology

The first step when tailoring an existing set of drivers is to clearly articulate the program's objectives. In addition, background information about the program is required to understand what the program is trying to accomplish and to gain an appreciation for its unique context and characteris-

tics. Meetings with program management and staff should be scheduled to collect all relevant background information.

After analysts gain a basic understanding of the program's context, they can then begin to tailor the drivers. Based on the objectives being assessed and the data that has been gathered, analysts must complete the following steps:

1. Determine which drivers do not apply to the program. Eliminate extraneous drivers from the set.
2. Establish whether any drivers are missing from the list. Add those drivers to the set.
3. Decide if multiple drivers from the set should be combined into a single, high-level driver. Replace those drivers with a single driver that combines them.
4. Decide if any drivers should be decomposed into multiple, more detailed drivers. Decompose each of those drivers into multiple drivers.
5. Adjust the wording of each driver to be consistent with the terminology and language of the program that is being assessed.

At this point, the tailored set of drivers can be used to assess the program's current state by completing the second activity of MOD Analysis, driver analysis.

5.2 Driver Analysis

The goal of driver analysis is to determine how each driver is influencing the objectives. More specifically, the probability of success state or failure state for each driver must be established. Notice that each driver question in Table 3 is expressed as a yes/no question that is phrased from the *success perspective*. Figure 7 depicts a driver question for the Security Process driver. This example will be used throughout this section when discussing driver analysis.

Driver 4: Security Process

Driver Question

Does the process being used to develop and deploy the system sufficiently incorporate security?

Considerations:

- Security-related tasks and activities in the program workflow
- Conformance to security process models
- Measurements and controls for security-related tasks and activities
- Process efficiency and effectiveness
- Software security development life cycle
- Security-related training
- Compliance with security policies, laws, and regulations

Response

- ❑ Yes
- ❑ Likely Yes
- ❑ Equally Likely
- ❑ Likely No
- ❑ No
- ❑ Don't Know

Figure 7: Driver Question and Range of Responses

Because the question in the figure is phrased from the success perspective, an answer of *yes* indicates the driver is in its success state and an answer of *no* indicates it is in its failure state. A range of answers is used to determine probabilities (likely yes, equally likely yes or no, likely no) when the answer is not a definitive yes or no. In addition, key items to consider when answering each question, called *considerations*, are provided for each driver question. The prototype set of stan-

dard driver questions for software security along with the considerations for each question are listed in Appendix A.

A set of *driver value criteria*, such as those shown in Figure 8, are normally used to support driver analysis. Driver value criteria serve two main purposes:

- They provide a definition of applicable responses to a driver question.
- They translate each response into the probability that the driver is in its success state, as well as the probability that it is in its failure state.

Response	Definition	Values	
		Probability of Success State	Probability of Failure State
Yes	The answer is almost certainly "yes." Almost no uncertainty exists. There is little or no probability that the answer could be "no." (~ > 95% probability of yes)	Maximum	Minimum
Likely yes	The answer is most likely "yes." There is some chance that the answer could be "no." (~ 75% probability of yes)	High	Low
Equally likely	The answer is just as likely to be "yes" or "no." (~ 50% probability of yes)	Medium	Medium
Likely no	The answer is most likely "no." There is some chance that the answer could be "yes." (~ 25% probability of yes)	Low	High
No	The answer is almost certainly "no." Almost no uncertainty exists. There is little or no probability that the answer could be "yes." (~ < 5% probability of yes)	Minimum	Maximum

Figure 8: Driver Value Criteria

The criteria for analyzing a driver must be tailored for each application of driver analysis. For example, the criteria in Figure 8 are based on a five-point scale, which allows decision makers to incorporate different levels of probability in their answers. A different number of answers (i.e., more or less than five) can be incorporated into the analysis when appropriate. In addition, some people prefer to include a response of *don't know* to highlight those instances where more information or investigation is needed before a driver can be analyzed appropriately.

When they analyze a driver, analysts need to consider how conditions and potential events[13] affect that driver. In general, the following items should be considered for each driver that is analyzed:

- positive conditions that support a response of *yes*
- negative conditions that support a response of *no*

[13] A *condition* is defined as the current state of being or existence. Conditions define an acquisition program's current circumstances. A *potential event* is defined as an occurrence or happening that alters an acquisition program's current conditions.

- potential events with positive consequences that support a response of *yes*
- potential events with negative consequences that support a response of *no*
- unknown factors that contribute to uncertainty regarding the response
- assumptions that might bias the response

Figure 9 shows an example of an analyzed driver. The answer to the driver question is *likely no*, which means that the driver is most likely in its failure state. As a result, the program's processes for security are most likely insufficient for achieving the objectives. The rationale for the response to each driver question should also be documented because it captures the reasons why analysts selected the response. Recording this information is important for historical purposes and for developing lessons learned.

Driver 4: Security Process

Driver Question

Does the process being used to develop and deploy the system sufficiently incorporate security?

Considerations:

- Security-related tasks and activities in the program workflow
- Conformance to security process models
- Measurements and controls for security-related tasks and activities
- Process efficiency and effectiveness
- Software security development life cycle
- Security-related training
- Compliance with security policies, laws, and regulations

Response

- ❑ Yes
- ❑ Likely Yes
- ❑ Equally Likely
- ☒ Likely No
- ❑ No
- ❑ Don't Know

Rationale

+ Previous acquisitions have a 90% history of delivering on-time.

− The process for integration testing is likely inadequate. Historically, integration testing has used informal agreements between vendors who had established working relationships. With this system, the vendors have not worked together previously, which makes informal agreements tenuous.

− The integration team is inexperienced. Many staff members are new to the program (45%).

− Training for new staff members is insufficient. Software security is not adequately addressed.

− The focus on security will require a significant change in vendors' standard processes. There is little evidence that people were properly trained in the new processes.

− QA did not have a chance to review the new processes before they were put into practice.

Figure 9: Analyzed Driver

5.3 Driver Profile

Finally, a *driver profile* provides a summary of all drivers relevant to the mission and objectives being assessed. A driver profile can be viewed as a dashboard that provides decision makers with a graphical summary of current conditions and expected performance in relation to the mission and objectives being pursued by a program. Figure 11 provides an example of a driver profile for

software security. In the figure, a bar graph is used to show 17 drivers that correspond to the standard set for software security.

The graph depicts the probability that each driver is in its success state. In addition, programmatic drivers are separated from the product drivers. The profiles in Figure 10 and Figure 11 indicate that the following four drivers are likely in their failure states: Security Process, Code Security, Operational System Security, and Security Risk Tolerance. These drivers should concern the program's decision makers.

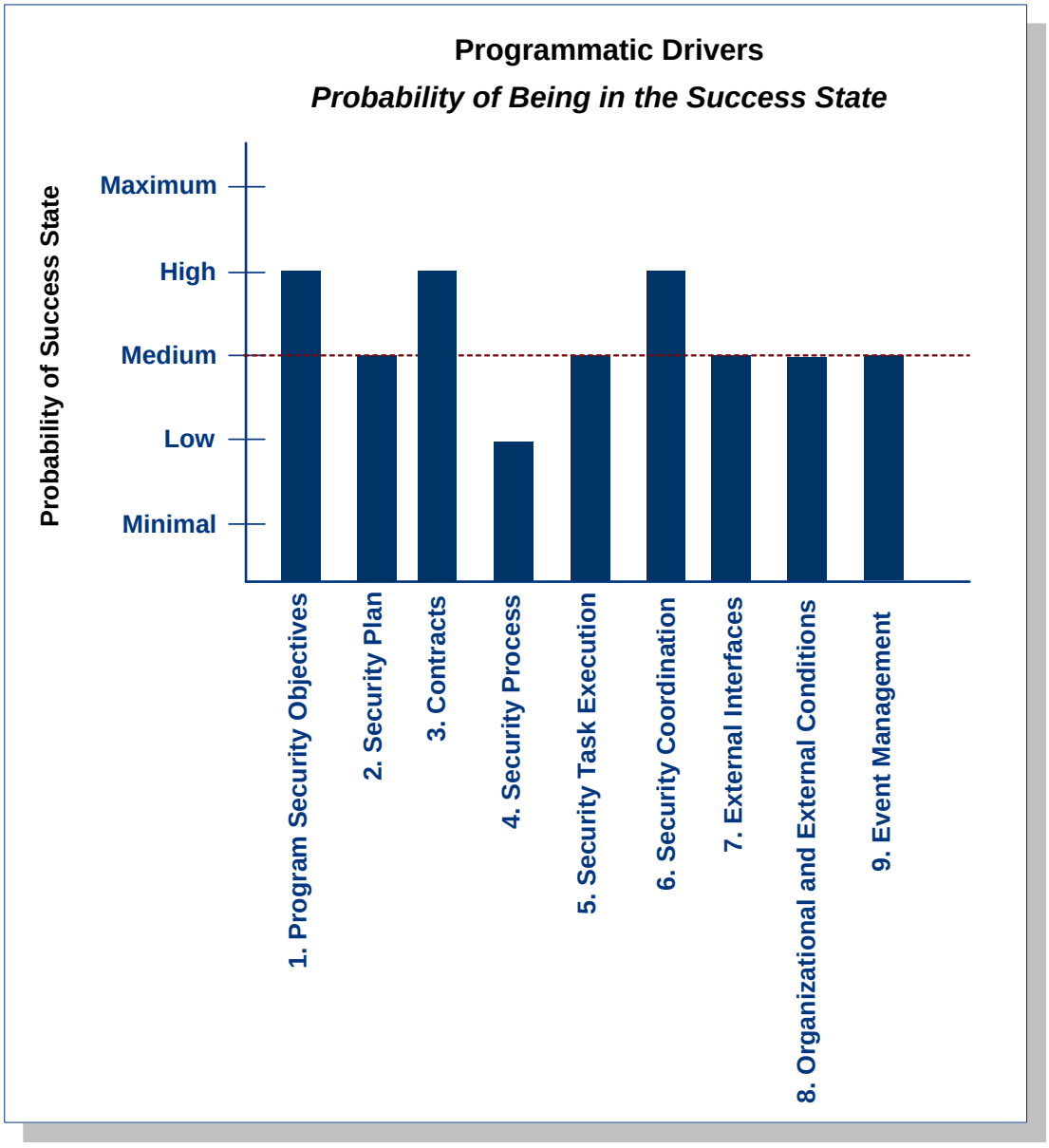

Figure 10: Driver Profile: Programmatic Drivers

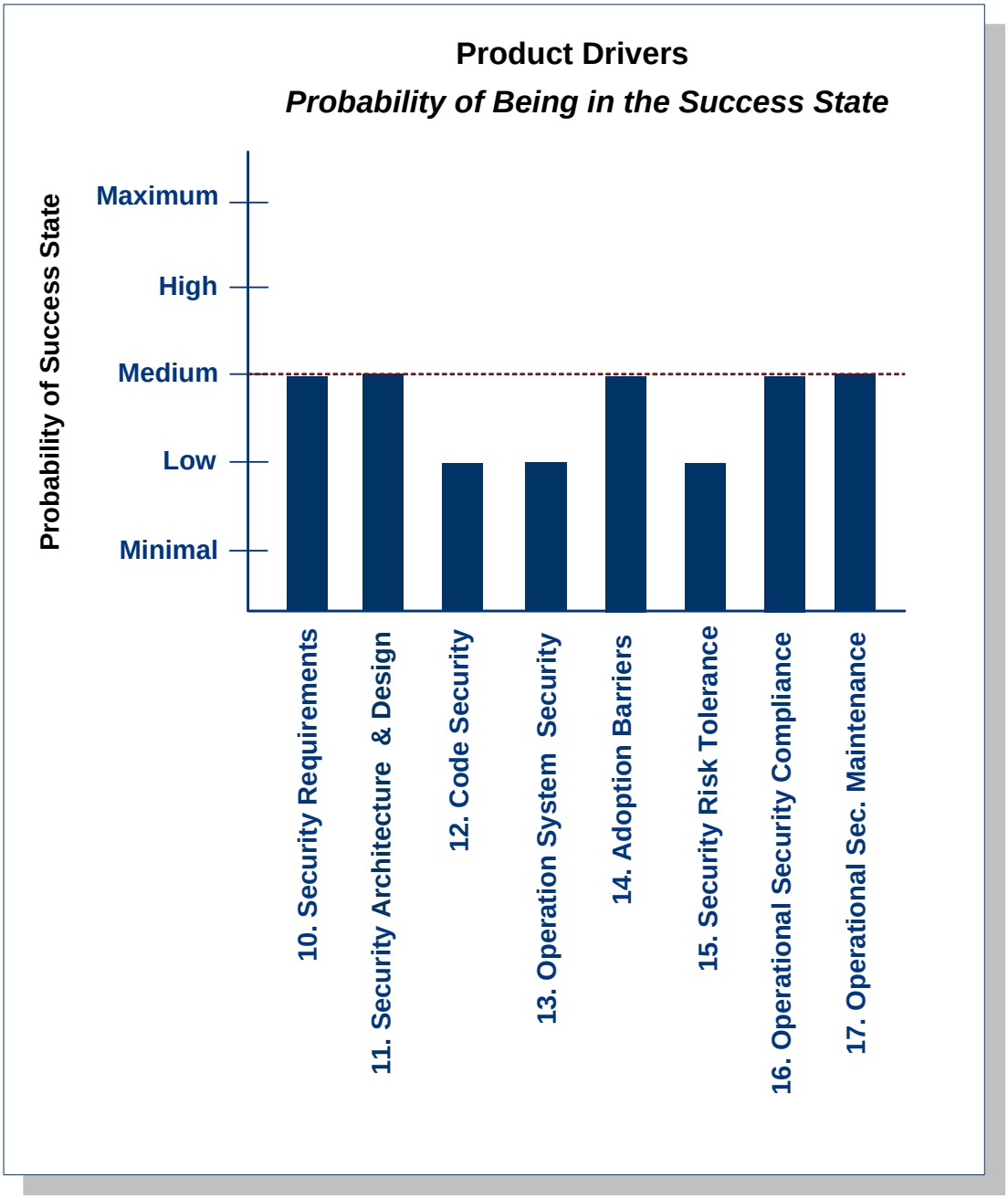

Figure 11: Driver Profile: Product Drivers

MOD Analysis enables systemic analysis of interactively complex socio-technical systems across the life cycle. As illustrated throughout this section, MOD Analysis defines an approach for assessing an acquisition program's ability to achieve its software security mission and objectives. It also forms the basis for a measurement and analysis framework for distributed management environments, which is described in the next section.

6 Integrated Measurement and Analysis Framework (IMAF)

The Integrated Measurement and Analysis Framework (IMAF) uses MOD Analysis to integrate subjective and objective data from multiple sources (targeted analysis, reports, and tactical measurement) and provide decision makers with a consolidated view of current conditions. To enable the construction of an integrated view of current conditions, the IMAF incorporates the two-tiered measurement and analysis approach presented earlier in Figure 5. (Refer to Section 4.2 for a description of the two-tiered measurement and analysis approach incorporating systemic analysis.)

Figure 12 below illustrates the basic structure of the IMAF. The fundamental objective of the framework is to provide decision makers with information about the performance of interactively complex socio-technical systems. The framework incorporates MOD Analysis as the approach for performing systemic analysis. As illustrated in Figure 12, MOD Analysis integrates data from a variety of sources, including targeted analysis, reports, and tactical measurement, and provides a common view of systemic measures to the decision maker. Because the IMAF incorporates systemic analysis, it can be used to measure performance in distributed management environments.

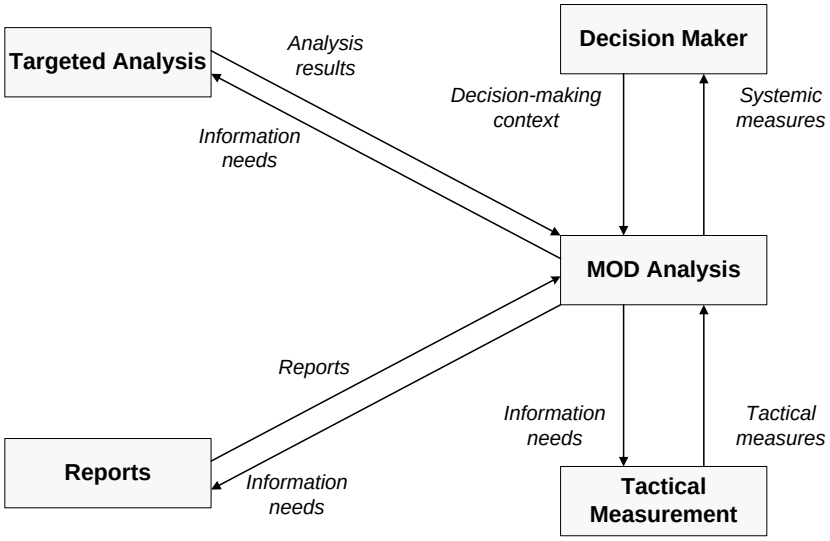

Figure 12: Integrated Measurement and Analysis Framework (IMAF)

The IMAF defines a systemic measure as the probability that a driver is in its success state. (Refer to Section 5.1.3 for a discussion of drivers.) A set of systemic measures (i.e., a driver profile) provides decision makers with insights into the overall performance of a socio-technical system. Systemic measures can be

- *qualitative*—categorized subjectively using expert judgment (e.g., maximum, high, medium, low, minimal)
- *quantitative*—derived using mathematical models (e.g., Bayesian Belief Networks[14])

[14] For information on Bayesian Belief Networks, refer to *Bayesian Inference in Statistical Analysis* by George Box and George Tiao, Wiley-Interscience, March 1992.

This section presents qualitative and quantitative implementations of the IMAF. First, Section 6.1 examines how qualitative systemic measures can be incorporated into a driver-based assessment approach. Next, Section 6.2 describes a quantitative implementation of the IMAF using Bayesian Belief Networks. Finally, Section 6.3 explores the link between tactical and systemic measurement and describes work being performed to align software security drivers (as described in Section 5.1.5) with existing software security codes of practice, such as the *Building Security In Maturity Model* [McGraw 2010].

6.1 Qualitative Implementation of the IMAF

Qualitative implementations of the IMAF can be used as the basis for conducting security assessments. In this report, an *assessment* is defined as a formal evaluation of a situation on behalf of stakeholders. The goal is to help decision makers understand (1) the current conditions affecting the situation and (2) what actions might be necessary to improve the status quo.

The quality of assessment results often correlates to the skills and abilities of the analysts who are conducting the assessment. The approaches underlying many assessments lack sufficient structure and definition, which means that analysts are not required to follow a defined set of activities when conducting the assessment. Each team of analysts conducting an assessment is free to follow whatever assessment protocol it deems appropriate for the given situation. The SEI is currently working to codify its software security assessments to make them more structured and repeatable. To achieve the goal of a structured and repeatable software security assessment, the SEI is beginning to develop an assessment approach that is based on a qualitative implementation of the IMAF and MOD Analysis. Figure 13 depicts the MOD Assessment, which incorporates qualitative analysis of drivers to establish the degree of confidence in mission success. (See Section 5.2 for a detailed discussion of driver analysis.)

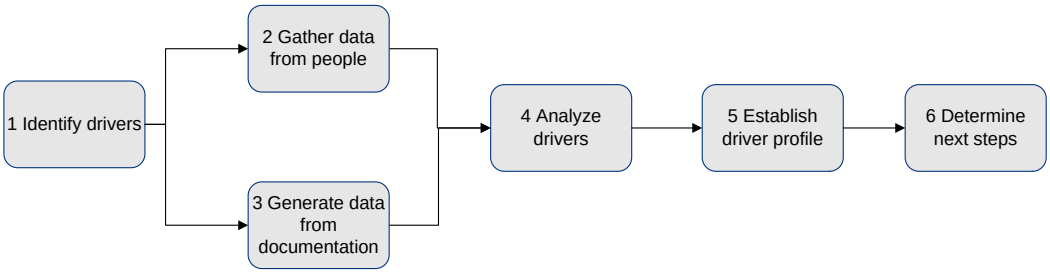

Figure 13: MOD Assessment

The activities of a MOD Assessment are as follows:

1. Identify drivers—Determine which drivers have a strong influence on the eventual outcome or result (based on mission and objectives). Refer to Section 5.1 for more information on identifying drivers.
2. Gather data from people—Elicit information about the program being assessed from people who play a role in executing it and then transform that information into usable data.
3. Generate data from documentation—Collect documentation relevant to the program (e.g., targeted analysis, reports, tactical measures) and generate usable data from that documentation.
4. Analyze drivers—Establish the probability that each driver is in its success state. Refer to Section 5.2 for more information on analyzing drivers.

5. Establish driver profile—Compile driver probabilities in a format that will support stakeholder decision making. Refer to Section 5.3 for more information on establishing a driver profile.
6. Determine next steps—Identify actions for maintaining or improving the current state.

Qualitative assessments have proven effective when evaluating operational security properties [Alberts 2002, GAO 1999]. The SEI's experience indicates that a qualitative approach should benefit stakeholders concerned with software security. However, in some instances stakeholders may want more quantitative measurement of software security if the required data is available. The next section presents one quantitative approach for implementing the IMAF.

6.2 Quantitative Implementation of the IMAF

As mentioned in Section 1, the purpose of measurement and analysis for researchers is to enable the testing of hypotheses while examining individual cause-and-effect or leading-indicator relationships. Such relationships eventually form a holistic model that can be used to predict the extent to which security objectives can be achieved.

Often, interviews with domain experts and/or a review of historical security data enable researchers to form assertions such as, "Rigorous code reviews increase confidence in meeting security objectives." A research hypothesis resulting from this assertion could be formed and documented more formally as follows:

- *H0:(Null Hypothesis)*: The rigor of code reviews has no relationship or correlation to the state of the Code Security Driver (refer to Appendix A, Driver 12).
- *H1:(Alternative Hypothesis)*: The rigor of code reviews has a significant positive relationship or correlation to the state of the Code Security Driver.

Armed with this formally stated hypothesis, the researcher can then sample observational data of code review rigor and the resulting Code Security Driver status, including observations of varying levels of code review rigor. By using accepted statistical hypothesis testing available in statistical tools and establishing acceptable alpha and beta statistical errors,[15] the researcher can then formulate a more compelling argument for the relationship or its absence.

Moving beyond simple hypothesis testing, there is a need to incorporate greater understanding of predictive modeling within the current body of knowledge related to security entities: security strengths, security weaknesses, opportunity and threat events, drivers, and objectives. Predictive analytics offers a basis for quantifying the likelihood of occurrence and relationships among these security entities. We believe that models based on predictive analytics will enable a more compelling and efficient basis for the research and use of these entities within the IMAF.

A variety of modeling approaches exist to quantitatively implement the IMAF, specifically providing a predictive analytics engine within the MOD Analysis depicted in Figure 11. These approaches include, but are not limited to

- traditional statistical correlation and regression analysis
- systems dynamics modeling

[15] Alpha error is the chance of incorrectly accepting the alternative hypothesis, while beta error is the chance of incorrectly accepting the null hypothesis.

- Monte Carlo simulation modeling
- probabilistic modeling (e.g., Bayesian Belief Networks)

After much discussion among the SEI researchers, the research project demands and constraints led to an initial modeling solution using Bayesian Belief Networks (BBNs). An example partial BBN for the software security objective is illustrated in Figure 14.

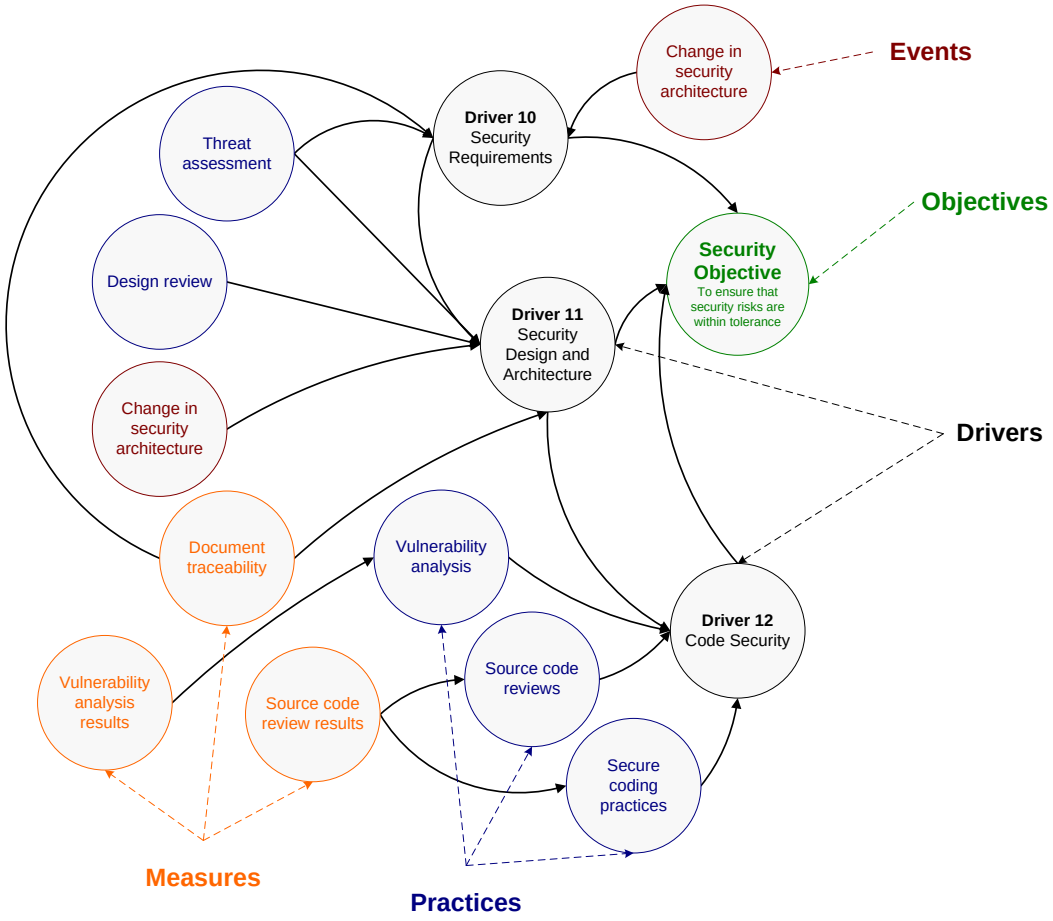

Figure 14: Notional Bayesian Belief Network Showing Security Entity Relationships with a Security Objective

Such a BBN would quantitatively confirm the likelihood of occurrence of specific security entities' states as well as confirm the relationships of "leading indicators" among the security entities. For example, in Figure 14, each of the entities, represented by circled nodes, have one or more states or conditions. These could be binary, such as success and failure, or they could be one of several levels on a scale of 1–5 indicating the rigor performed. Additionally, in Figure 14, each arrow represents a potential cause-and-effect relationship or leading indicator relationship. Three security drivers are selected as those that primarily explain the believed status of the security objective. Likewise, the status of driver 11 (Security Design and Architecture) may be determined in advance with knowledge of the following entities:

- driver 10 (Security Requirements)
- threat assessment

- design review
- a change in the security architecture
- document traceability measure

Analysis may demonstrate that some relationships that reflect conventional wisdom are not significant, while other relationships may be newly determined. Consequently, even though Figure 14 represents subjective expert opinion that the three identified drivers provide insight into the status of the security objective, the empirical analysis may show that only two of these drivers are statistically or probabilistically linked to the security objective. In this case, the BBN is modified and the nonsignificant relationship is dropped from the model.

Finally, an operational BBN model, which learns from additional experience and data, will prove useful for identifying security practices that provide the strongest contribution to meeting the security objective. Referring back to the example in Figure 14, the team acquires additional objective data or subjective security expert opinion on entities and their relationships. The model is constructed to learn from this and is easily updated in terms of state probabilities within each entity and the relationships between entities.

In addition to supporting decision making in operational settings, the BBN also serves as a research platform, expressing what is known and not known in entity relationships. It helps identify when additional new entities are needed and in which areas of the overall model. The key evaluation of BBN model sufficiency rests on how narrow a probability distribution of the "effect" entity node results from the observations of the "causal" or "leading-indicator" nodes: too wide a probability distribution decreases the practicality of using the BBN.

The specific aspects of BBN modeling that address both the operational and research uses of the IMAF and MOD Analysis include the following (all of the illustrative examples refer to Figure 13):

- Ability to gain some analytical freedom from many of the statistical assumptions behind classical statistical methods (e.g., not being limited to statistical regression to explain relationships between security entities in the BBN model)
- ability to create a holistic, quantitative model of the entire set of security drivers
- ability not only to forecast the future but to explain past events that most likely led to today's situation. For example, if we have observations of the status of drivers 10 and 12, we can begin to make a forecast of the likely status of the security objective. We can also render our best judgment about the status of driver 11 and the code review, assuming we were not privy to these two entities for this particular organization.
- ability to analyze and model both objective and subjective data, thus capitalizing on subject matter expert judgment. For example, we may only have actual observation data on three of the drivers and the design and code reviews, and the remainder of the entities would be based on subjective expert opinion until we could collect observed values of those entities. The observed values would come from actually observing the entity state, such as whether or not a change in the security architecture has occurred or the degree of change in the security architecture.

- ability to operate and make predictions with incomplete data. For example, we may not be able to observe the status of several of the drivers such as driver 12 (Code Security) and driver 11 (Security Design and Architecture). Even in this situation, the BBN is able to use accepted conditional probabilistic computations to update all of the unobserved entities in the model. For example, although drivers 12 and 11 are unobserved, our observations of the other entities will still enable us to update our forecast of the security objective and also conclude the most likely state of drivers 12 and 11, simply based on the observed nodes all around them. In summary, we can forecast forward or diagnose backward using the BBN to update the calculations.
- ability to make predictions for situations or scenarios not experienced before. For example, it might be the case that as we populate the model, we find that several of the scenarios of the combined driver states have not been experienced before. In this situation, the BBN model will use accepted probabilistic computations to forecast the status of the security objective.
- ability to incorporate a learning mechanism similar to artificial intelligence and neural network applications. For example, we may learn over time that the states within a given entity, such as driver 12 (Code Security), may actually be occurring with different probabilities than previously thought. The BBN model can learn and update these probabilities as new experience and data become available—similar to how email spam filters learn over time to more effectively reject unwanted email using a Bayesian mechanism.

Using BBNs to implement the IMAF shows considerable promise for providing decision makers with quantitative measurement data. From a decision maker's perspective, BBNs offer an approach to model real-time observations and update predictions with the latest knowledge, thereby providing decision makers with current and comprehensive information before making critical decisions. The BBN model (as shown in Figure 14) is based upon the mapping of missions to objectives to drivers to practices to measures. The next section examines how to align drivers to software security codes of practice.

6.3 Aligning Drivers with Software Security Codes of Practice

Performing meaningful measurement and analysis based on carefully considered and defined software software security measures derives from a clear statement of the mission or purpose for an acquisition program. This statement is further expanded into a set of objectives that reflect the mission (refer to Section 5.1.2). In Section 5, this report describes the concept of drivers and the process for determining a set of drivers that best reflect the program objectives for software security. Given a set of well-defined drivers that reflect objectives, how do we then derive meaningful measures that increase our confidence that the software functions (or will function) securely in operation? And how do we connect measures to leading software security codes of practice (frameworks and models) that are being used in organizations today to build more secure software?

Figure 15 depicts this relationship. Eventually, as part of the IMAF, we intend to determine if we can define a measurement and analysis method for identifying a set of practices that are more effective in satisfying software security objectives.

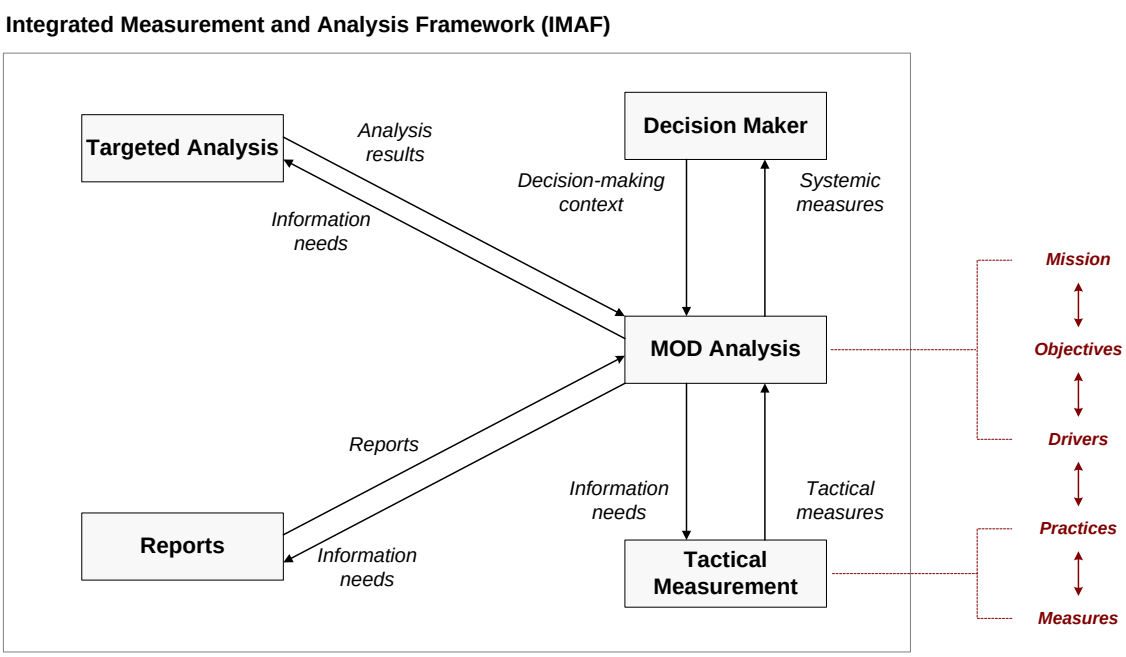

Figure 15: Aligning Drivers with Software Security Codes of Practice

The codes of practice that we have reviewed thus far include

- Building Security In Maturity Model (BSIMM): http://bsimm2.com/
- Open Web Application Security Project (OWASP) Software Assurance Maturity Model (SAMM): http://www.owasp.org/index.php/Category:Software_Assurance_Maturity_Model
- Microsoft Secure Development Lifecycle: http://www.microsoft.com/security/sdl/
- Department of Homeland Security Measurement work and Assurance for CMMI Process Reference Model: https://buildsecurityin.us-cert.gov/swa/
- CERT Resilience Management Model, Resilient Technical Solution Engineering Process Area: http://www.cert.org/resilience/rmm.html

Examples of the current state of the practice for expressing measures for software security include the following, taken from the OWASP SAMM, version 1.0 [OpenSAMM 2009]:

- >90 percent applications and data assets evaluated for risk classification in past 12 months
- >60 percent development staff trained within past 1 year
- >80 percent staff certified within past 1 year
- >50 percent of projects with updated attack surface analysis in past 12 months
- >50 percent of projects with updated security requirements design-level analysis in past 12 months
- >50 percent of project teams performing code review on high-risk code in past 6 months

While these implementation measures are informative, they describe only the presence, absence, and extent of some action. They say nothing about the contribution or effectiveness of such actions toward achieving a desired state of software security to meet a defined program objective.

During this phase of our research, we defined both a top-down (Part 1) and a bottom-up (Part 2) process for aligning drivers with software security codes of practice (and vice versa) to produce meaningful measures. The Part 1 process uses an abbreviated Goal Question (Indicator) Metric (GQIM) Method developed at the SEI [Park 1996]. The GQIM draws upon earlier work done by Basilli and Rombach [Basili 1984, 1988, 1994; Rombach 1989] in defining a Goal Question Metric (GQM) method. For both of these approaches and for goal-driven measurement in general, the primary question is not "What metrics should I use?" but "What do I want to know or learn?" [SEI 2010].

These two processes are described in Table 4.

Table 4: Align Drivers with Software Security Codes of Practice

Approach	Step	Step Description
Top Down (Part 1)	1.1	Select a driver of interest.
	1.2	Determine what information items and measures are needed to answer the driver question (and inform driver considerations).
	1.3	Categorize measures as one of the following types: • Implementation: Is this activity/process being performed (presence, absence, extent)? • Effectiveness: How good is the work product or outcome of the activity/process? Does it achieve the intended result? • Process performance: Is the process performing as expected? Is the process defined, documented, planned, managed, monitored, controlled, and continuously improved?
	1.4	Determine if the measures appear in (or can be derived from) one or more codes of practice.
	1.5	If this analysis provides useful insights: 1. Develop a traceability matrix that shows driver/code of practice relationships. 2. Repeat for all drivers that support a given objective. 3. Meet with domain experts to establish quantification schemes for each practice-based measure. 4. Use predictive analytics methods (such as BBNs) to determine the most useful drivers, measures, and practices.
Bottom Up (Part 2)	2.1	Select a practice area and companion practices of interest.
	2.2	Determine which objective information needs the measures from this practice area might meet.
	2.3	Determine if the information needs to map to one or more drivers.
	2.4	If this analysis provides useful insights: 1. Develop a traceability matrix that describes this relationship. 2. Analyze one code of practice completely to see if ○ practices suggest new or modified drivers ○ a practice area does not map to any driver and is therefore (perhaps) of lower priority or less optimal for meeting program objectives than others

We have conducted Steps 1.1, 1.2, and 1.3 of the top-down analysis (Part 1 in Table 4) for the software security drivers shown in Figure 14 (drivers 10, 11, and 12). In this report, we present the result of the top-down analysis for driver 10:

Driver 10: Security Requirements

Do requirements sufficiently address security?

Considerations:

1. Customer, user, and stakeholder security requirements and needs
2. Tradeoffs between security, performance, and other quality attributes
3. Operational security requirements
4. Information security requirements
5. Maturity of technology used and implications for security requirements
6. Relevant policies, standards, guidelines, and regulations
7. Results of risk analysis of security requirements
8. Analysis of security threats as they affect security requirements (using methods such as misuse/abuse cases, threat models, and attack patterns)
9. Process for developing security requirements

One derivation of practices and measures that support answering the question for driver 10 (Security Requirements) appears in Table 5 below.

Table 5: Part 1 Example for Driver 10 Security Requirements: Do Requirements Sufficiently Address Security?

Category	Practice	Measure	Notes
Implementation Measures	Product security requirements are documented.	▪ % of software products for which security requirements are/are not documented	Maps to considerations 1, 3, 4, and 6
	There is traceability between all sources of security requirements and each security requirement; each security requirement has at least one source.	▪ % of security requirements with/without identified sources	Maps to considerations 1, 3, 4, and 6
	Maturity of technologies used is determined and documented.	▪ % of in-use technologies with/without designated maturity ratings	Maps to consideration 5
	Tradeoff analysis results are documented.	▪ % of security requirements with/without documented tradeoff analyses	Maps to consideration 2
	Security requirements risk analysis results are documented. Actions to address and/or mitigate high-priority risks are resolved in a timely manner.	▪ % of security requirements with/without documented risk analysis results ▪ Elapsed time to close actions for high-priority risks associated with security requirements ▪ Elapsed time to implement mitigation actions for high-priority risks associated with security requirements	Maps to consideration 7
	Security threat analysis results and methods are documented. Actions to address high-priority analysis results are resolved in a timely manner.	▪ % of security requirements with/without documented threat analysis results ▪ Elapsed time to resolve actions for high-priority threat analysis results associated with security requirements	Maps to consideration 8
	There is a defined process used to elicit, prioritize, review, and document security requirements.	▪ % of security requirements that have/have not been elicited, prioritized, reviewed, and documented as a result of executing a defined, documented process	Maps to consideration 9
Effectiveness Measures	Product security requirements adequately address customer, user, and stakeholder requirements and needs.	▪ % of security requirements that meet (do not meet) customer-, user-, and stakeholder-defined thresholds for adequacy	Need further expansion of how adequacy is determined
	Product security requirements adequately address • tradeoff analyses • operational and information security requirements • maturity of technology • policies, standards, guidelines, and regulations • security risk analysis results • threat analysis results	• % of security requirements that have/have not been determined to address stakeholder-defined thresholds for adequacy for all of the bulleted items listed in the practice description	Need further expansion of how adequacy is determined
Process Performance Measures	The process used to specify security requirements performs as expected.	▪ Extent to which the defined process for specifying security requirements meets its performance criteria	Need to define process performance criteria

Steps 1.4 and 1.5 of the top-down analysis (Part 1 in Table 4) and all steps in the bottom-up analysis (Part 2 in Table 4) will be performed in FY11. The drivers, practices, and measures of interest will be informed by pilot driver assessment results and BBN modeling. We made a conscious decision not to take this driver/practice/measure analysis further until we had access to data to help guide our selection of meaningful drivers and practices.

The final section of this report describes next steps and future directions for this research project.

7 Insights and Next Steps

This section summarizes our research results for FY10, presents some surprises and insights experienced by the authors thus far, and identifies our research directions for FY11.

7.1 Summary

The goal of the software security measurement and analysis research project is to address the following three research questions:

- Q1: How do we establish, specify, and measure justified confidence that a software product[16] is sufficiently secure to meet operational needs?
- Q2: How do we measure at each phase of the development or acquisition life cycle that the required/desired level of security has been achieved?
- Q3: How do we scale measurement and analysis approaches to complex environments, such as large, distributed systems of systems?

As described in this report, in FY10 we have focused on addressing research question 1, which establishes the foundation for the remaining questions. We have defined an approach, illustrated by an example, for addressing research question 2. We have yet to determine our plans and approach for examining research question 3.

This report presents a foundational set of measurement concepts, including a generic measurement process. It describes how meaningful measures serve to provide the information that decision makers need when they need it and in the right form. Decision makers are those responsible for developing and acquiring software-reliant systems. The distributed management environment is the prevailing circumstance faced by today's decision makers. Performance in these environments, including software security performance, is better analyzed using a two-tiered approach that incorporates system decomposition and event analysis as well as systemic analysis. Mission-Objective-Driver Analysis is one method for conducting systemic analysis. The SEI has developed a starter set of drivers for software security, and this report presents how these can be further analyzed, tailored, and assessed. The report closes by defining the Integrated Measurement and Analysis Framework (IMAF) and methods for qualitatively and quantitatively collecting data to inform this framework. Such data is used to analyze software security performance. The closing section also introduces work in progress to address research question 2.

7.2 Surprises and Insights

This section includes a few insights and surprises experienced by the authors in comparing where we started to what we have learned in the course of conducting this research over the past seven months. We also speculate a bit as to what we may learn going forward.

When we originally conceived of this effort, we intended to evaluate current practice (as represented by codes of practice for software security) and see what common themes emerged.

[16] For example, a software application, set of applications, software-reliant system, or system of systems.

We also knew we would need to establish a method for defining what constituted adequate security and then propose an approach for demonstrating this throughout the life cycle. Our initial (albeit limited) thinking was to use the work in assurance cases as the foundation.

One early insight was that we had a rich body of existing SEI knowledge and research results to help frame a robust method for expressing the definition of adequate security. This informed driver identification, driver analysis, and the IMAF. Assurance case research results can certainly contribute to this expression and be used for collecting evidence of adequate security, but it is only one of several methods.

Another insight was that we could consider more than just a single organization, single project, and single software-reliant system and begin to address systems of systems in distributed management environments. Of course, this ability to scale from small systems to large systems and from single systems to systems of systems has yet to be demonstrated, but we were able to develop hypotheses about it.

We were surprised by the lack of data related to software security. Much of what people know about software security seems to be based on conventional wisdom. We do see some anecdotal evidence backing up claims about practice, but there is little objective data. In the long term, we believe that our work on the IMAF will be able to provide objective data about what drives software security. This data can then be used to support or refute claims about practice effectiveness.

We were also surprised by our community's inability to effectively articulate what constitutes success for software security. Our nonsecurity field experience has shown that program managers traditionally struggle when defining what constitutes success for their programs (i.e., defining objectives). However, they seem even less prepared to define software security objectives. We believe the results of this research project have real potential to establish solid, quantitative benchmarks for software security.

This project represents one of the CERT Program's first opportunities to tap into the extensive knowledge and experience of the SEI's Software Engineering Measurement and Analysis team. The team enabled us to identify predictive analytics and leverage the power of Bayesian Belief Networks to provide insights regarding the use of quantitative methods. The BBN method's ability to inform while tolerating uncertainty, incomplete and missing data, and subjective data (very much the current state of practice in software security) is very promising.

That said, we believe we may find in the predictive analytics work that much of today's conventional wisdom regarding drivers and leading indicators for software security may very well not pan out. This type of debunking is common as we seek empirical data and quantitative significance of relationships. We may also find some new factors as leading indicators that we had not thought of before.

We believe we will experience some of the key concepts espoused by Douglas Hubbard [Hubbard 2007] and Nassim Nicholas Taleb [Taleb 2010]:

1. More information is not always better and can often hurt the decision-making situation.
2. There are valid approaches to trading off the value of additional information against the cost of collecting it, specifically Hubbard's Applied Information Economics (AIE) method.

3. Measurement error, especially from human sources, may be far greater than we believed and may represent a major issue that must be solved before modeling can occur. Hubbard offers promising approaches to better eliciting expert opinions and conducting assessments.

7.3 Research Directions

In FY11, we plan to tackle the following tasks:

- Seek community feedback and improvement suggestions on this research project and this report.
- In the pursuit of research question 1, seek pilot assessment data and other types of data-gathering opportunities to improve and validate the IMAF.
- Define a starter Bayesian Belief Network in support of one software security objective and its related drivers. Use assessment and data-gathering opportunities to begin populating this network. Introduce the use of automated tools for expressing and analyzing BBNs.
- Conduct the two-part process for addressing research question 2 (refer to Section 6.3). Analyze the leading software security codes of practice while performing this process.
- Explore the applicability of measurement and analysis methods in general and in related disciplines, for example, Douglas Hubbard's methods [Hubbard 2007] and how they inform this work.
- Describe the differences and similarities between this work and model-based measurement and analysis work, such as the Capability Maturity Modeling® Integration for Development (CMMI®-DEV) [CMMI Product Team 2006] and the CERT® Resilience Management Model (CERT®-RMM) [Caralli 2010]. We believe that the set of software security drivers may serve as a useful point of connection/intersection.
- Develop materials (graphics, brochures, presentations, white papers, etc.) that provide simplified explanations of complex measurement methods and aid us in communicating about this work, both internally and externally.

® Capability Maturity Modeling, CMMI, CERT Resilience Management Model, and CERT®-RMM are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

Appendix A: Standard Set of Drivers for Software Security

This appendix provides a prototype set of driver questions for assessing software security as described in Section 5.1.5. This set of drivers is derived from the following software security objective: *When the system is deployed, security risks to the deployed system will be within an acceptable tolerance*.

At this point in time, the set of drivers has not been validated through the SEI's field-piloting activities. The next step in the development of the drivers is to validate them through field testing. Once the set of drivers is validated, it can serve as an archetype that analysts can use for tailoring purposes.

Programmatic Drivers

1. ***Program Security Objectives***: Are the program's security objectives realistic and achievable?
2. ***Security Plan***: Does the plan for developing and deploying the system sufficiently address security?
3. ***Contracts***: Do contract mechanisms with partners, collaborators, subcontractors, and suppliers sufficiently address security?
4. ***Security Process***: Does the process being used to develop and deploy the system sufficiently incorporate security?
5. ***Security Task Execution***: Are security-related tasks and activities performed effectively and efficiently?
6. ***Security Coordination***: Are security activities within the program coordinated appropriately?
7. ***External Interfaces***: Do work products from partners, collaborators, subcontractors, or suppliers meet security requirements?
8. ***Organizational and External Conditions***: Are organizational and external conditions facilitating completion of security tasks and activities?
9. ***Event Management***: Is the program able to identify and manage potential events and changing circumstances that affect its ability to meet its software security objectives?

Product Drivers

10. ***Security Requirements***: Do requirements sufficiently address security?
11. ***Security Architecture and Design***: Do the architecture and design sufficiently address security?
12. ***Code Security***: Is the code sufficiently secure?
13. ***Operational System Security***: Is the integrated system sufficiently secure to support operations?

14. ***Adoption Barriers***: Have barriers to customer/user adoption of the system's security features been managed appropriately?
15. ***Operational Security Compliance***: Will the system comply with applicable security policies, laws, and regulations?
16. ***Operational Security Preparedness***: Are people prepared to maintain the system's security over time?
17. ***Security Risk Tolerance***: Is the security risk of the deployed system within an acceptable tolerance?

Driver 1: Program Security Objectives

Are the program's security objectives realistic and achievable?

Considerations:

1. Tradeoffs between security and performance
2. Funding allocated to developing security features and controls
3. Schedule allocated to developing security features and controls
4. Maturity of technology used and implications for security
5. Resources dedicated to security tasks and activities
6. Availability of staff with security experience and expertise
7. Appropriate security credentials of staff

Driver 2: Security Plan

Does the plan for developing and deploying the system sufficiently address security?

Considerations:

1. Security-related tasks and activities in the program plan
2. Funding allocated to developing security features and controls
3. Schedule allocated to developing security features and controls
4. Resources dedicated to security tasks and activities
5. Availability of staff with security experience and expertise
6. Appropriate security credentials of staff
7. Security-related training
8. Definition of security roles and responsibilities

Driver 3: Contracts

Do contract mechanisms with partners, collaborators, subcontractors, and suppliers sufficiently address security?

Considerations:

1. Contracts with each partner, collaborator, subcontractor, and supplier
2. Alignment among the contracts of all partners, collaborators, subcontractors, and suppliers
3. Security-related tasks and activities outlined in contracts
4. Partner or collaborator resources dedicated to security tasks and activities
5. Mechanisms for ensuring compliance with the terms and conditions of contracts
6. Contingency or business continuity plans to deal with significant noncompliance issues

Driver 4: Security Process

Does the process being used to develop and deploy the system sufficiently incorporate security?

Considerations:

1. Security-related tasks and activities in the program workflow
2. Conformance to security process models
3. Measurements and controls for security-related tasks and activities
4. Process efficiency and effectiveness
5. Software security development life cycle
6. Security-related training
7. Compliance with security policies, laws, and regulations

Driver 5: Security Task Execution

Are security-related tasks and activities performed effectively and efficiently?

Considerations:

1. Experience and expertise of management and staff
2. Availability of staff with security experience and expertise
3. Staff knowledge of software security methods, tools, and technologies
4. Security-related training
5. Experience with the software security development life cycle
6. Methods, tools, and technologies supporting secure development
7. Compliance with security policies, laws, and regulations

Driver 6: Security Coordination

Are security activities within the program coordinated appropriately?

Considerations:

1. Communication of security information
2. Dependencies of security tasks and activities
3. Relationships among partners, collaborators, subcontractors, and suppliers
4. Alignment of security roles defined in contracts
5. Alignment of security plans across all participants
6. Interoperability of security processes across all participants
7. Interoperability of methods, tools, and technologies across all participants
8. Compliance with security policies, laws, and regulations
9. Programmatic complexity

Driver 7: External Interfaces

Do work products from partners, collaborators, subcontractors, or suppliers meet security requirements?

Considerations:

1. Security of applications
2. Security of software
3. Security of systems or subsystems
4. Security of hardware
5. Risk of malicious code in work products
6. Risk of vulnerabilities in work products
7. Potential for realization of security risks in deployed system
8. Compliance with security policies, laws, and regulations

Driver 8: Organizational and External Conditions

Are organizational and external conditions facilitating completion of security tasks and activities?

Considerations:

1. Stakeholder sponsorship of software security (e.g., sponsors, funders, developers, customers, users)
2. Strategies and actions of upper management related to software security
3. Effect of policies, laws, and regulations on security
4. Effects of relationships with partners, collaborators, subcontractors, and suppliers on security
5. Effects of external (world, marketplace) conditions (technological, environmental, economic, political) on security

Driver 9: Event Management

Is the program able to identify and manage potential events and changing circumstances that affect its ability to meet its software security objectives?

Considerations:

1. Expected and unexpected potential events and changing circumstances
2. Changes in security requirements
3. Changes in security architecture
4. Changes in security methods, tools, or technologies
5. The effects of staff or supplier changes on security
6. The effects of changes in partnerships, agreements, or contracts on security
7. Schedule slack for developing security features and controls
8. Funding reserve for developing security features and controls
9. Risk management plan, process, and tools
10. Risk mitigation plans
11. Program continuity, disaster, and contingency plans

Driver 10: Security Requirements

Do requirements sufficiently address security?

Considerations:

1. Customer, user, and stakeholder security requirements and needs
2. Tradeoffs between security, performance, and other quality attributes
3. Operational security requirements
4. Information security requirements
5. Maturity of technology used and implications for security requirements
6. Relevant policies, standards, guidelines, and regulations
7. Results of risk analysis of security requirements
8. Analysis of security threats as they affect security requirements (using methods such as misuse/abuse cases, threat models, and attack patterns)
9. Process for developing security requirements

Driver 11: Security Architecture and Design

Do the architecture and design sufficiently address security?

Considerations:

1. Software and system architecture
2. Tradeoffs between security, performance, and other quality attributes
3. Maturity of technology used and implications for security architecture and design
4. Security of COTS products
5. Security requirements including information and operational security requirements
6. Analysis of security threats as they affect architecture and design (using methods such as misuse/abuse cases, threat models, and attack patterns)
7. Results of risk analysis of the security architecture and design
8. Results of security vulnerability analysis
9. Process for addressing security as part of software/system architecture and design
10. Architecture and design reviews

Driver 12: Code Security

Is the code sufficiently secure?

Considerations:

1. Security requirements
2. Security architecture and design
3. Secure coding standards, guidelines, and practices
4. Analysis of common vulnerabilities
5. Source code reviews
6. Static code analysis
7. Software security testing
8. Security issues addressed during unit testing
9. Code interfaces and dependencies as they affect security
10. Process for secure coding
11. Analysis of security threats as they affect code (using methods such as misuse/abuse cases, threat models, and attack patterns)
12. Analysis of common weaknesses
13. Complexity of code
14. Results of risk analysis of the secure code

Driver 13: Operational System Security

Is the integrated system sufficiently secure to support operations?

Considerations:

1. Operational security requirements
2. Vulnerability analysis
3. Security risk analysis
4. Software security testing
5. Penetration testing
6. Effectiveness of security features and controls
7. Security issues addressed during integration testing
8. Security of legacy systems
9. Security of COTS software
10. Disaster recovery, contingency, and business continuity plans

Driver 14: Adoption Barriers

Have barriers to customer/user adoption of the system's security features been managed appropriately?

Considerations:

1. Stakeholder sponsorship of security
2. Changes to operational workflows
3. Tradeoffs between performance and security
4. Training provided to users, operators, and maintainers
5. Support provided to users, operators, and maintainers

Driver 15: Operational Security Compliance

Will the system comply with applicable security policies, laws, and regulations?

Considerations:

1. Security-related policies affecting the operational environment
2. Security-related laws affecting the operational environment
3. Security-related regulations affecting the operational environment
4. Security-related standards of care affecting the operational environment

Driver 16: Operational Security Preparedness

Are people prepared to maintain the system's security over time?

Considerations:

1. Patching and upgrades
2. Supplier support for security features and controls in custom software
3. Supplier support for COTS software
4. Supplier support for legacy systems
5. Degree of changes to the system
6. Unanticipated use of the system
7. Operational policies, practices, and procedures for maintaining operational systems over time
8. Operational staffing levels
9. Experience and expertise of operators, maintainers, and users
10. Security-related training for operators, maintainers, customers, and users
11. Physical security in operational environment
12. Ongoing mitigation of operational security risks
13. Periodic vulnerability analysis and penetration testing
14. Disaster recovery, contingency, and business continuity plans

Driver 17: Security Risk Tolerance

Is the security risk of the deployed system within an acceptable tolerance?

Considerations:

1. Maturity of technology used and implications for security
2. Security of COTS products
3. Effectiveness of security processes, practices, and controls
4. Security issues addressed during integration testing
5. Security issues addressed during system testing
6. Operational (i.e., post-deployment) security policies, practices, and procedures
7. Risk of malicious code in deployed system
8. Risk of vulnerabilities in deployed system
9. Potential for realization of security risks
10. Acceptable mitigation of risk (considering users, customers, stakeholders, and others who may be impacted by the system)
11. Business continuity, disaster recovery, and contingency plans
12. Some articulation of what constitutes an acceptable (or unacceptable) level of risk
13. Process for managing risk (identify, analyze, mitigate)

Appendix B: Related Sources and Efforts

The following sources and efforts provide foundational work and additional details on the state of the practice in measuring and analyzing cyber security, information security and assurance, and software security and assurance. These sources have informed and will continue to inform this research project.

- *Practical Measurement Framework for Software Assurance* [Bartol 2008]. This report 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.
- *Measuring Cyber Security and Information Assurance: State-of-the-Art Report (SOAR)* [Bartol 2009]. This report assesses the current state of the art in cyber security and information assurance measurement to facilitate further research into this subject.
- *Performance Measurement Guide for Information Security: Special Publication 800-55 Revision 1* [Chew 2008]. Section 3.5.2 includes a discussion of application security measurement.
- The Center for Internet Security Consensus Information Security Metrics [CIS 2010]. See specifically metrics related to application security.
- *Acquisition Measurement Version 1.0* [Creel 2008]. This report provides a foundation for the discussion and advancement of acquisition measurement. Its primary focus is on improving the way in which acquisition projects and organizations manage and conduct their activities. A secondary focus is on management and oversight of the supplier.
- U.S. Department of Homeland Security Software Assurance Forum Measurement Working Group Presentations. They are available at https://buildsecurityin.us-cert.gov/swa/forums.html.
- U.S. Department of Homeland Security Build Security In website measurement articles, including “Measures and Measurement for Secure Software Development.” Articles are available at https://buildsecurityin.us-cert.gov/bsi/articles/best-practices/measurement.html.

Glossary

acquisition program
A collection of individual projects that work collaboratively to provide technical capabilities via a set of networked software products (i.e., a system of systems).

condition
The current state of being or existence. Conditions define an acquisition program's current circumstances.

distributed management environment
Multiple, independently managed organizational entities working collaboratively to achieve a common mission or purpose.

driver
A factor that has a strong influence on the eventual outcome or result (i.e., whether or not objectives will be achieved).

measure
A value assigned to a variable that is used to provide a decision maker with insight into a given characteristic or property of an entity.

mission
The fundamental purpose of an individual, group, or operation that defines the target, or focus, of the measurement and analysis effort.

objective
A tangible outcome or result that must be achieved when pursuing a mission.

organizational entity
A social unit of people that pursues collective goals, controls its own performance, shares a single senior manager, and operates under a single set of policies.

potential event
An occurrence or happening that alters an acquisition program's current conditions.

program
See *acquisition program*.

project
A planned set of interrelated tasks that are executed over a fixed period of time and within certain constraints, such as cost or funding limitations. Unlike operations, which are repetitive and permanent (or semipermanent) functional tasks intended to produce products or provide services, projects are temporary in nature.

socio-technical system
Interrelated technical and social elements that are engaged in goal-oriented behavior. Elements of a socio-technical system include the people who are organized in teams or departments to do their work tasks and the technical systems on which people rely when performing work tasks.

software project
An endeavor that is intended to produce a software product, such as a software application or software-reliant system, within a fixed period of time and within specified budget constraints.

system decomposition and event analysis
An analysis approach in which a socio-technical system's critical components are evaluated for potential failures.

system of systems
a set or arrangement of interdependent systems that are related or connected (i.e., networked) to provide a given capability [Levine 2003]. The following characteristics are used to differentiate a system of systems from a very large, complex monolithic system [Maier 1996]:

- *managerial independence*—The management of each system within a system of systems is independent from the management of the other systems.
- *operational independence*—Each system within a system of systems provides useful functionality apart from other systems.
- *evolutionary character*—Each system within a system of systems grows and changes independently of other systems over time.
- *emergent behavior*—Certain behaviors of a system of systems arise from the interactions among the individual systems and are not embodied in any of the individual systems.
- *geographic distribution*—Individual systems within a system of systems are dispersed over large geographic areas.

system theory
A multidisciplinary analysis approach that considers a system as comprising a series of distinct but interconnected components or subsystems.

systemic analysis
An approach in which a system is analyzed as a whole rather than decomposing it into individual components and then analyzing each component separately; systemic analysis is based on system theory.

systemic measures
A measure that provides a decision maker with insight into the overall performance of a socio-technical system.

tactical measures
A measure that provides a decision maker with insights into a specific task that must be performed or some characteristic of a work product.

References

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

[Alberts 2002]
Alberts, Christopher & Dorofee, Audrey. *Managing Information Security Risks: The OCTAVE Approach*. Addison-Wesley, 2002.
http://www.sei.cmu.edu/library/abstracts/books/0321118863.cfm

[Alberts 2009]
Alberts, Christopher & Dorofee, Audrey. *A Framework for Categorizing Key Drivers of Risk* (CMU/SEI-2009-TR-007). Software Engineering Institute, Carnegie Mellon University, 2009.
http://www.sei.cmu.edu/library/abstracts/reports/09tr007.cfm

[Bartol 2008]
Bartol, Nadya. *Practical Measurement Framework for Software Assurance and Information Security*. Practical Software and Systems Measurement Support Center, 2008.
http://www.psmsc.com/Prod_TechPapers.asp

[Bartol 2009]
Barton, Nadya; Bates, Brian; Goertzel, Karen M.; & Winograd, Theodore. *Measuring Cyber Security and Information Assurance: State-of-the-Art Report (SOAR)*. Information Assurance Technology Analysis Center, 2009. http://iac.dtic.mil/iatac/reports.jsp

[Basili 1984]
Basili, Victor R. & Weiss, David M. "A Methodology for Collecting Valid Software Engineering Data." *IEEE Transactions on Software Engineering, SE-10*, 6 (November 1984): 728-738.
http://www.cs.umd.edu/~basili/publications/journals/J23.pdf

[Basili 1988]
Basili, Victor R. & Rombach, H. Dieter. "The TAME Project: Towards Improvement-Oriented Software Environments." *IEEE Transactions on Software Engineering 14*, 6 (June 1988): 758-773.

[Basili 1994]
Basili, Victor R.; Caldiera, Gianluigi; & Rombach, H. Dieter. "The Goal Question Metric Approach." *Encyclopedia of Software Engineering*. Wiley, 1994.

[Caralli 2010]
Caralli, Richard A.; Allen, Julia H.; Curtis, Pamela D.; White, David W.; & Young, Lisa R. *CERT® Resilience Management Model, v1.0* (CMU/SEI-2010-TR-012). Software Engineering Institute, Carnegie Mellon University, 2010.
http://www.sei.cmu.edu/library/abstracts/reports/10tr012.cfm

[Chew 2008]
Chew, Elizabeth; Swanson, Marianne; Stine, Kevin; Bartol, Nadya; Brown, Anthony; & Robinson, Will. *Performance Measurement Guide for Information Security: Special Publication 800-55 Revision 1*. National Institute of Standards and Technology (NIST), 2008. http://csrc.nist.gov/publications/nistpubs/800-55-Rev1/SP800-55-rev1.pdf. Additional NIST reports are available at http://csrc.nist.gov/publications/PubsSPs.html.

[CIS 2010]
The Center for Internet Security (CIS). *CIS Consensus Information Security Metrics*. http://cisecurity.org/en-us/?route=downloads.metrics (2010).

[CISWG 2005]
Corporate Information Security Working Group (CISWG). *Report of the Best Practices and Metrics Teams*. United States House of Representatives, 2005. http://www.educause.edu/Resources/CorporateInformationSecurityWo/153504

[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

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

[Creel 2008]
Creel, Rita; Dean, Joe; & Jones, Cheryl. *Acquisition Measurement Version 1.0*. Practical Software and Systems Measurement Support Center, 2008. http://www.psmsc.com/Prod_TechPapers.asp

[Dorofee 2008]
Dorofee, Audrey; Marino, Lisa; & Alberts, Christopher. *Lessons Learned Applying the Mission Diagnostic* (CMU/SEI-2008-TN-004). Software Engineering Institute, Carnegie Mellon University, 2008. http://www.sei.cmu.edu/library/abstracts/reports/08tn004.cfm

[GAO 1999]
United States General Accounting Office (GAO). *Information Security Risk Assessment: Practices of Leading Organizations* (GAO/AIMD-00-33). GAO, 1999. http://www.gao.gov/special.pubs/ai00033.pdf

[Hubbard 2007]
Hubbard, Douglas W. *How to Measure Anything: Finding the Value of "Intangibles" in Business*. John Wiley & Sons, 2007.

[ISO 2007]
International Organization for Standardization. *ISO/IEC 15939:2007, Systems and Software Engineering – Measurement Process*, 2nd ed. ISO, 2007.

[Leveson 2004]
Leveson, Nancy. "A New Accident Model for Engineering Safer Systems." *Safety Science 42*, 4 (April 2004): 237-270. http://sunnyday.mit.edu/accidents/safetyscience-single.pdf

[Levine 2003]
Levine, Linda; Meyers, B. Craig; Morris, Ed; Place, Patrick R. H.; & Plakosh, Daniel. *Proceedings of the System of Systems Interoperability Workshop (February 2003)* (CMU/SEI-2003-TN-016). Software Engineering Institute, Carnegie Mellon University, 2003. http://www.sei.cmu.edu/reports/03tn016.pdf

[Maier 1996]
Maier, M. "Architecting Principles for Systems-of-Systems," 567-574. *Proceedings of the Sixth Annual International Symposium of INCOSE*. Boston, MA, July 7-11, 1996. INCOSE, 1996.

[McGarry 2002]
McGarry, John; Card, David; Jones, Cheryl; Layman, Beth; Clark, Elizabeth; Dean, Joseph; & Hall, Fred. *Practical Software Measurement: Objectives for Decision Makers*. Addison-Wesley, 2002.

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

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

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

[Park 1996]
Park, Robert; Goethert, Wolfhart; & Florac, William. *Goal-Driven Software Measurement —A Guidebook* (Handbook CMU/SEI-96-HB-002). Software Engineering Institute, Carnegie Mellon University, 1996.

[Rombach 1989]
Rombach, H. Dieter & Ulery, Bradford T. "Improving Software Maintenance Through Measurement." *Proceedings of the IEEE 77*, 4 (April 1989): 581-595.

[SEMA 2009]
Software Engineering Measurement and Analysis (SEMA) Group. *Measurement and Analysis Infrastructure Diagnostic (MAID) Evaluation Criteria, Version 1.0* (CMU/SEI-2009-TR-022). Software Engineering Institute, Carnegie Mellon University, 2009. http://www.sei.cmu.edu/library/abstracts/reports/09tr022.cfm

[SEI 2010]
Software Engineering Institute. *Measurement & Analysis*. http://www.sei.cmu.edu/measurement (2010).

[Stamatis 2003]
Stamatis, D. H. *Failure Mode and Effect Analysis: FMEA from Theory to Execution*, 2nd revised and expanded ed. ASQ Quality Press, 2003.

[Taleb 2010]
Taleb, Nassim Nicholas. *The Black Swan*, 2nd ed. Random House, 2010.

[WEIS 2010]
The Ninth Workshop on the Economics of Information Security (WEIS 2010). Harvard University, June 2010. http://weis2010.econinfosec.org/index.html. Agendas and papers presented from prior WEIS events are available online.

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
Integrated Measurement and Analysis Framework for Software Security	FA8721-05-C-0003

6. AUTHOR(S)

Christopher Alberts, Julia Allen, Robert Stoddard

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-TN-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	

11. SUPPLEMENTARY NOTES

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

13. ABSTRACT (MAXIMUM 200 WORDS)

In today's business and operational environments, multiple organizations routinely work collaboratively to acquire, develop, deploy, and maintain technical capabilities via a set of interdependent, networked systems. Measurement in these distributed management environments can be an extremely challenging problem. The CERT® Program, part of Carnegie Mellon University's Software Engineering Institute (SEI), is developing the Integrated Measurement and Analysis Framework (IMAF) to enable effective measurement in distributed environments, including acquisition programs, supply chains, and systems of systems. The IMAF defines an approach that integrates subjective and objective data from multiple sources (targeted analysis, reports, and tactical measurement) and provides decision makers with a consolidated view of current conditions. This report is the first in a series that addresses how to measure software security in complex environments. It poses several research questions and hypotheses and presents a foundational set of measurement concepts. It also describes how meaningful measures provide the information that decision makers need when they need it and in the right form. Finally, this report provides a conceptual overview of the IMAF, describes methods for qualitatively and quantitatively collecting data to inform the framework, and suggests how to use the IMAF to derive meaningful measures for analyzing software security performance.

14. SUBJECT TERMS	15. NUMBER OF PAGES
software security, measurement, measure, distributed environment, systemic analysis, Bayesian Belief Network	76

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