Software Engineering Institute

Risk-Based Measurement and Analysis: Application to Software Security

Christopher Alberts
Julia Allen
Robert Stoddard

February 2012

TECHNICAL NOTE
CMU/SEI-2012-TN-004

CERT® Program

http://www.sei.cmu.edu

Carnegie Mellon

This material is based upon work funded and supported by United States Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center sponsored by the United States Department of Defense.

This report was prepared for the

Contracting Officer
ESC/CAA
20 Shilling Circle
Building 1305, 3rd Floor
Hanscom AFB, MA 01731-2125

Table of Contents

List of Figures

List of Tables

Acknowledgments

The authors thank Archie Andrews, Carol Woody, and Dave Zubrow for their sponsorship and support of this work. We thank Rita Creel and Audrey Dorofee for their careful technical review of this document and their thoughtful comments. We also thank Audrey Dorofee for her technical contribution to the development of the Mission Risk Diagnostic as part of the SEI's Mission Success in Complex Environments (MSCE) special project. We would also like to thank Alexa Huth for editing this document. Finally, the authors thank the SEI's CERT® Program for providing the funding to conduct this research effort.

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

Abstract

For several years, the software engineering community has been working to identify practices aimed at developing more secure software. Although some foundational work has been performed, efforts to measure software security assurance have yet to materialize in any substantive fashion. As a result, decision makers (e.g., development program and project managers, acquisition program offices) lack confidence in the security characteristics of their software-reliant systems. The CERT® Program at Carnegie Mellon University's Software Engineering Institute (SEI) has chartered the Software Security Measurement and Analysis (SSMA) Project to advance the state-of-the-practice in software security measurement and analysis. The SSMA Project is exploring how to use risk analysis to direct an organization's software security measurement and analysis efforts. The overarching goal is to develop a risk-based approach for measuring and monitoring the security characteristics of interactively complex software-reliant systems across the lifecycle and supply chain. To accomplish this goal, the project team has developed the SEI Integrated Measurement and Analysis Framework (IMAF) and refined the SEI Mission Risk Diagnostic (MRD). This report is an update to the technical note, *Integrated Measurement and Analysis Framework for Software Security* (CMU/SEI-2010-TN-025), published in September 2010. This report presents the foundational concepts of a risk-based approach for software security measurement and analysis and provides an overview of the IMAF and the MRD.

1 Introduction

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 answer. However, meaningful measurement is about transforming strategic direction, policy, and other forms of management decision into action and measuring the performance of that action.

Effective 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 basic goal of measurement and analysis is to provide decision makers with the information they need, when they need it, and in the right form. In recent years, researchers have begun to turn their attention to the topic of software security assurance and how to measure it.

Software security assurance is justified confidence that software-reliant systems are adequately planned, acquired, built, and fielded with sufficient security to meet operational needs, even in the presence of attacks, failures, accidents, and unexpected events. For several years, various groups within the software engineering community have been working diligently to identify practices aimed at developing more secure software. However, efforts to measure software security assurance have yet to materialize in any substantive fashion, although some foundational work has been performed.

As a result of the software engineering community's interest, the CERT® Program at Carnegie Mellon University's Software Engineering Institute (SEI) chartered the Software Security Measurement and Analysis (SSMA) Project in October 2009 to advance the state-of-the-practice related in software security measurement and analysis. The SSMA Project builds on the CERT Program's core competency in software and information security as well as the SEI's work in software engineering measurement and analysis. The purpose of this new research project is to address the following two questions:

1. How do we establish, specify, and measure justified confidence that interactively complex software-reliant systems are sufficiently secure to meet operational needs?
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?

In essence, the two research questions examine how decision makers (for example, development program and project managers as well as acquisition program officers) can measure and monitor the security characteristics of interactively complex software-reliant systems across the life cycle and supply chain. This report is primarily focused on answering the first research question.

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

1.1 Technical Approach

To answer the first research question, we are proposing to use risk analysis as a means of directing an organization's software security measurement and analysis efforts. This concept is shown in Figure 1. Consider the specific example where the decision maker is an acquisition program manager. From a software security perspective, the program manager wants to establish a reasonable degree of confidence that the software product being acquired and developed will be sufficiently secure to meet operational needs. In other words, the program manager is interested in establishing some benchmark of software security assurance.

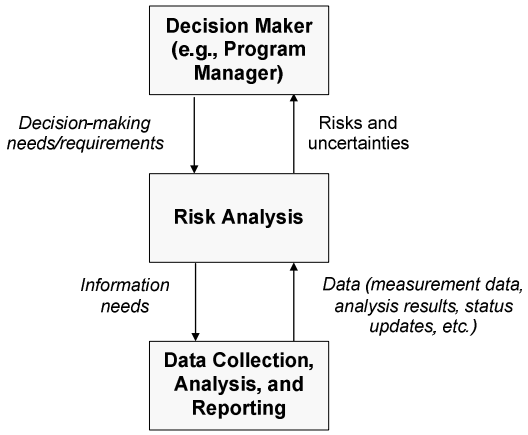

Figure 1: Risk-Based Decision Making

Risk analysis is one approach that can be used to establish software security assurance for a software product. If the security risk to the deployed software product is kept within an acceptable tolerance, then the manager will have a reasonable degree of confidence that the software product is sufficiently secure to meet operational needs (i.e., reasonable assurance). An inverse relationship exists between risk and assurance: As risk is reduced, the degree of assurance increases proportionally (and vice versa).

Figure 1 shows that risk analysis provides the program manager with an understanding of the program's current risks and uncertainties. A risk analysis can provide the manager with an indication of whether or not the program is on track for success. However, uncertainties reflect circumstances where there are known gaps in the underlying data or where the data collected are not fully trusted. As a result, uncertainties provide the program manager with an opportunity to collect additional data in order to reduce the degree of decision-making uncertainty inherent in the current situation.

The program manager can then update his or her decision-making needs or requirements based on the goal of reducing uncertainty. The decision-making needs or requirements are then translated into revised information needs that are used to identify additional data that need to be collected. These data can be collected using a variety of mechanisms, including assessments, status reporting, and measurement. Over time, the reduction in uncertainty resulting from new data that are collected, analyzed, and reported should provide decision makers with more clarity regarding system performance. As a result, the reduction in uncertainty enables better decision making based on more objective data.

1.2 Audience

Our primary audiences for this technical report are measurement program implementers and measurement researchers. Managers who are responsible for overseeing software acquisition and development programs will find information in this report to be helpful. In addition, people interested in software and security measurement and analysis or process improvement will also find useful information in this report.

1.3 Structure of this Report

This technical report is an update to the technical note, *Integrated Measurement and Analysis Framework for Software Security* (CMU/SEI-2010-TN-025), published in September 2010 [Alberts 2010]. This report includes the following sections:

- *Section 2: Measurement Concepts*—provides an overview of key measurement and analysis concepts and describes a measurement process
- *Section 3: Two Approaches for Analyzing Risk*—presents an overview of concepts underlying tactical risk analysis and systemic risk analysis
- *Section 4: Mission Risk Diagnostic (MRD)*—describes an approach for conducting systemic risk analysis of interactively complex software-reliant systems
- *Section 5: Integrated Measurement and Analysis Framework (IMAF)*—presents a framework that uses risk and uncertainty to direct measurement and analysis activities
- *Section 6: Additional Research Tasks*—describes three additional research tasks addressed by the SSMA research project: measure identification, standard mapping, and driver modeling
- *Section 7: Summary and Next Steps*—summarizes the key content of the report and highlights next steps for the SSMA project
- *Appendix: Standard Set of Drivers for Software Security*—presents the standard set of 17 drivers for software security

Overall, the goal of this report is to describe a risk-based approach for establishing, specifying, and measuring justified confidence that interactively complex software-reliant systems are sufficiently secure to meet operational needs. The next section provides the foundation for achieving this goal by presenting key measurement concepts.

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 the SSMA research project and this report. Measurement and analysis involves gathering quantitative data about products, processes, and projects and analyzing that data to influence actions and plans. Measurement and analysis activities allow decision makers to achieve the following outcomes [Park 1996, SEI 2010]:

- **characterize**, to gain an understanding of processes, products, resources, and environments and to establish baselines for comparisons with future assessments
- **evaluate**, to determine the current status with respect to plans
- **predict**, by understanding relationships among processes and products and building models of these relationships, so that the values observed for some attributes can be used to predict others
- **improve**, by identifying roadblocks, root causes, inefficiencies, and other opportunities for improving product quality and process performance

Many definitions for the term *measurement* exist. For this project, we have adopted the following definition: a set of observations that reduce uncertainty where the result is expressed as a quantity [Hubbard 2007]. For measurement to have an impact, it must affect the behavior of decision makers. If decisions are not influenced by measurement activities, then measurement provides no added value [Hubbard 2007].

A process for measurement and analysis defines, implements, and sustains a measurement capability, ensuring that the information needs of decision makers are satisfied. 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 (e.g., a global supply chain).

Measurement activities and their relationships are shown in Figure 2, 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]. An effective measurement process, such as the one illustrated in Figure 2, exhibits the following characteristics [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 are 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 identified through evaluation and use of the measurement process and measures are communicated to the measurement process owner.

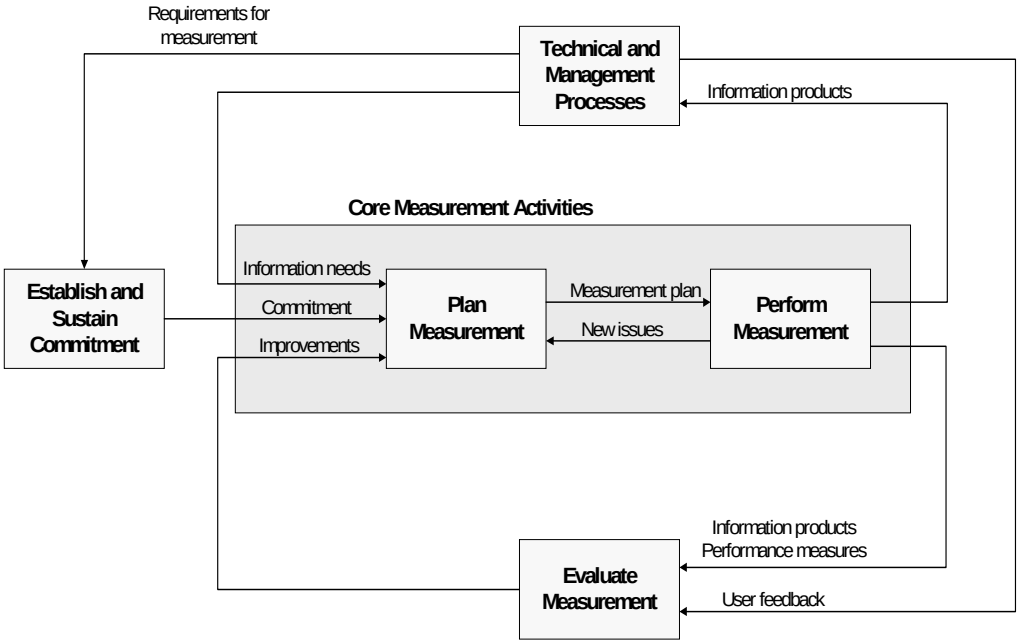

Figure 2: Measurement Process

Our research agenda maps to the core measurement activities depicted in Figure 2 (*plan measurement* and *perform measurement*). In Section 1 of this report, we highlighted the two questions that we intend to answer when conducting this research project. The first question is: How do we establish, specify, and measure justified confidence that interactively complex software-reliant systems are sufficiently secure to meet operational needs? This question maps to the *plan measurement* activity from Figure 2. This report is primarily focused on answering this question by describing an approach for planning measurement activities.

Our second research question is: How do we measure at each phase of the development or acquisition life cycle that the required/desired level of security has been achieved? Question two is focused on how to conduct measurement activities during each lifecycle phase. As a result, question two maps to the *perform measurement* activity of Figure 2. While our current work only touches upon the second research question, our future research and development activities will focus on addressing this research question and describing an approach for performing measurement activities. As our research project progresses, we intend to address all four measurement-related activities from Figure 2 (establish and sustain commitment, plan measurement, perform measurement, and evaluate measurement).

The measurement and analysis activities depicted in Figure 2 are briefly described in the remainder of this section so that *plan measurement* and *perform measurement* are presented in the context of the full process.[1]

2.1 Establish and Sustain Commitment

Measurement and analysis cannot succeed without management and stakeholder commitment, both up front as the measurement and analysis process is being scoped and defined and on an ongoing basis as the process is implemented. Stakeholder commitment requires a sponsor who ensures that decision makers and key stakeholders are fully engaged. The sponsor works with measurement stakeholders to

- allocate the resources necessary to execute all process activities on a sustaining basis
- use the measurement reports that result from the process
- 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 barriers to a comprehensive measurement and analysis effort. Individuals and project groups must view measurement as a positive and purposeful activity that is 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.

2.2 Plan Measurement

The *plan measurement* activity encompasses (1) the identification of information needs for decision makers and (2) the selection and definition of appropriate measures to address those needs. As defined in this report, a *measure* is a variable to which a value is assigned as the result of measurement [ISO 2007]. 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

[1] A detailed description of each measurement and analysis activity is available in *ISO/IEC 15939: 2007* [ISO 2007] and *Practical Software Measurement* [McGarry 2002].

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.

2.3 Perform Measurement

The *perform measurement* 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. Analysis and reporting includes formulating recommendations for decision makers and providing alternative courses of action based on measurement results.

2.4 Evaluate Measurement

The *evaluate measurement* activity assesses both the measures that are used, as well as the capability of the measurement process itself. It ensures that the measurement approach is continually updated to address the information needs of decision makers as well as to promote an increasing maturity of the measurement process.

The quality of measurement data is particularly important. Poor quality data can lead to incorrect assumptions and bad decisions, which can erode people's trust in the measurement data that are collected. As a result, the quality and effectiveness of all information products produced by the measurement process must be evaluated using predefined criteria.

Evaluating a measurement process ultimately leads to the identification of improvements to the measurement effort. The measurement process may be evaluated in the following four ways [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 evaluation 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

Improving the measurement process involves a wide variety of solutions based on identified deficiencies. Improvements can range 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[2] (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] http://en.wikipedia.org/wiki/Ishikawa_diagram

2.5 Technical and Management Processes

A fifth activity, *technical and management processes*, is shown in Figure 2. While this activity is not a measurement-oriented activity, it is important for setting the context within which measurement is conducted. As illustrated in Figure 2, technical and management processes interface directly with the measurement process. Decision makers use their knowledge of technical and management processes to define information needs that are used to direct measurement activities. In addition, decision makers consume information products provided by the measurement process to support the decisions they make when managing technical and management processes.

To this point in the report, we have focused on the foundational concepts of measurement and analysis. In the next section, we expand the foundation by presenting two basic approaches for conducting risk analysis: tactical risk analysis and systemic risk analysis.

3 Two Approaches for Analyzing Risk

Our research is focused on developing risk-based approaches for measuring and analyzing the performance of interactively complex software-reliant systems across the life cycle and supply chain. To fully appreciate what this statement means, you need to understand the phrase, "interactively complex software-reliant systems."

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 technologies on which people rely when performing work tasks. Projects, programs, and operational processes are all examples of socio-technical systems. A *software-reliant system* is a socio-technical system whose behavior (e.g., functionality, performance, safety, security, interoperability, and so forth) is dependent on software in some significant way [Bergey 2009]. In the remainder of this document, when we use the word *system*, we are referring to a software-reliant system.

Interactive complexity refers to the presence of unplanned and unexpected sequences of events in a system that are either not visible or not immediately understood [Perrow 1999]. The components in an interactively complex system interact in relatively unconstrained ways. When a system is interactively complex, independent failures can interact with the system in ways that cannot be anticipated by the people who design and operate the system.

Measurement and analysis should be tailored to the context in which it will be applied. In our research project, we have been focused on using risk analysis to direct the measurement and analysis of interactively complex systems. Two distinct risk analysis approaches can be used when evaluating systems: (1) tactical risk analysis and (2) systemic risk analysis.[3]

3.1 Tactical Risk Analysis

Risk is the probability of suffering harm or loss. From the tactical perspective, risk is defined as the probability that an event will lead to a negative consequence or loss. The basic goal of tactical risk analysis is to evaluate a system's components for potential failures. Tactical risk analysis is based on the principle of system decomposition and component analysis. The first step of this approach is to decompose a system into its constituent components. The individual components are then prioritized, and a subset of components is designated as being critical. Next, the risks to each critical component are analyzed.

Tactical risk analysis enables stakeholders to (1) determine which components are most critical to a system and (2) analyze ways in which those critical components might fail (i.e., analyze the risk to critical components). Stakeholders can then implement effective controls designed to mitigate those potential failures. Because of its focus on preventing potential failures, tactical risk analysis has been applied extensively within the discipline of systems engineering. However, analysts need

[3] The discussion of tactical and systemic risk analysis is adapted from "A New Accident Model for Engineering Safer Systems" [Leveson 2004].

to understand the limitations of using tactical risk analysis to evaluate interactively complex systems, which include the following:

- Only critical components are analyzed. Non-critical components are not examined, and interdependencies among components are not addressed.
- The selection of which conditions and events (i.e., sources or causes of risk) to consider is subjective.
- Non-linear relationships among conditions and events (e.g., feedback) are not considered. Risk causal relationships are presumed to be simple, direct, and linear.
- 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.
- Confidence in the performance of individual components does not establish confidence in the performance of the parent system.

In addition, when you attempt to decompose interactively complex systems, some system-wide behaviors become lost. It is very difficult to establish the relationship between the macro-level behavior of the system and the micro-level behavior of individual components. As a result, tactical risk analysis provides a partial picture of the risks to an interactively complex system. To get a more holistic view of risk in an interactively complex system, you need to employ an alternative analysis approach.

3.2 Systemic Risk Analysis

From the systemic perspective, risk is defined as the probability of mission failure (i.e., not achieving key objectives). Systemic risk, also referred to as mission risk in this document, examines the aggregate effects of multiple conditions and events on a system's ability to achieve its mission. Systemic risk analysis is based on system theory. The underlying principle of system theory is to analyze a system 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 risk analysis thus provides a holistic view of the risk to an interactively complex socio-technical system. The first step in this type of risk analysis is to establish the objectives that must be achieved. The objectives define the desired outcome, or "picture of success," for a 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 systemic factors, called *drivers* in this report, are important because they define a small set of factors that can be used to assess a 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 overall risk to the system's mission.

Applying systemic risk analysis to interactively complex systems provides decision makers with a means of confidently assessing the behavior of the system as a whole, which is necessary when assessing assurance. The next section of this report builds on the concepts outlined in this section by describing a method for analyzing systemic risk in interactively complex systems.

4 Mission Risk Diagnostic (MRD)[4]

The SEI is developing the Mission Risk Diagnostic (MRD) to enable systemic risk analysis of interactively complex systems. During our research and development activities over the past few years, we demonstrated how the MRD provides an efficient and effective means of analyzing risk in interactively complex systems, such as acquisition programs [Alberts 2009, Dorofee 2008].[5] Our current work builds on this initial research by exploring how to apply the MRD in a software security context. When the MRD is applied in this context, the target of the risk analysis is the project[6] or a program[7] that is acquiring and developing a software product (e.g., an integrated software-reliant system).[8] The goal is to gauge whether the security risk of the deployed software product will be within an acceptable tolerance.

The following two tasks form the foundation of the MRD: (1) driver identification and (2) driver analysis. This section describes both tasks in detail, presents a discussion of the driver profile, which is the main output of the MRD, introduces the concept of mission risk, and concludes with a description of the MRD's key tasks and steps. The concepts and examples presented in this section are described in the context of a large-scale acquisition and development program, which is one specific type of interactively complex system.

4.1 Driver Identification

The main goal of driver identification is to establish 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 established, 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.

4.1.1 Mission

The MRD defines the term *mission* as the fundamental purpose of the system that is being examined. 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 [Alberts 2009]. The following

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

[5] The MRD builds off of and expands on the work of the SEI Mission Success in Complex Environments (MSCE) Special Project. For more information on MSCE, see http://www.sei.cmu.edu/risk/.

[6] In this document, the term *project* is defined as a planned set of interrelated tasks to be executed over a fixed period of time and within certain cost and other limitations.

[7] In this document, the term *program* is defined as a group of related projects managed in a coordinated way to obtain benefits and control not available from managing them individually. Programs usually include an element of ongoing activity.

[8] In this example, the project or program is an example of a software-reliant system because it comprises one or more groups of people that rely on software technologies when performing work tasks. Similarly, the product that is acquired, developed, and deployed in this example is also a software-reliant system.

is an example of a mission statement as required by the MRD: *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 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.

4.1.2 Objectives

In the MRD, an *objective* is defined as a tangible outcome or result that must be achieved when pursuing a mission [Alberts 2009]. 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. 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 the MRD 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.
- 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*.[9] 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

[9] This objective is focused on whether the tactical security risks affecting a deployed, operational system will be within an acceptable tolerance. Tactical risk analysis is commonly used to mitigate operational security risks when acquiring, engineering, and developing a technology. In this section, the MRD is being used to predict whether or not the tactical security risks of a deployed, operational system will be within an acceptable tolerance. Here, a systemic risk analysis approach (the MRD) is being used early in the life cycle (during development) to predict the results of a tactical risk analysis that will be performed later in the life cycle (during operations). For more information on tactical and systemic risk analysis, see Section 3 of this document.

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 the MRD 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 MRD. The remainder of this section describes a qualitative implementation of the MRD. We are also working on quantitative implementation of the MRD, which we intend to present in other reports.[10]

4.1.3 Drivers

The MRD 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) [Alberts 2009]. Table 1 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 1 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.

Table 1: 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

The above list can be used define a qualitative scale for driver analysis. Analyzing each driver in relation to the qualitative scale establishes a benchmark of performance in relation to a system's documented mission and objectives.

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

4.1.4 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 3. When dealing with multiple objectives, analysts must be sure to record these relationships to enable effective decision making.

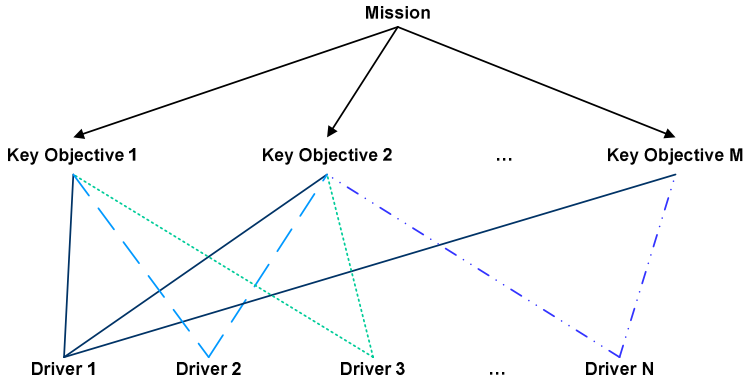

Figure 3: Relationships among Objectives and Drivers

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.

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 10–25 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 have been developed by SEI researchers.

4.1.5 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. (More details about the 17 drivers can be found in the appendix section of this report.) Table 2 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 4.1.2 and have not been validated in pilot assessments.[11] 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 2: 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.	Integrated System Security	Does the integrated system sufficiently address security?
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.	Product Security Risk Management	Is the approach for managing product security risk sufficient?

The drivers in Table 2 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 a system (e.g. an acquisition program) is being managed. Drivers 10-17 are referred to as *product* drivers because they provide insight into the product that is being acquired, developed, and deployed.

4.1.6 Tailoring an Existing Set of Drivers

The standard drivers (Table 2) describe general security concerns that analysts should consider when assessing the security characteristics of software products being developed and deployed by

[11] The standard set of software security drivers were derived from the following objective: When the system is deployed, security risks to the deployed system will be within an acceptable tolerance.

acquisition programs. 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 characteristics.

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 conducting driver analysis.

4.2 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 2 is expressed as a yes/no question that is phrased from the *success perspective*. Figure 4 depicts a driver question for the Security Process driver. This example will be used throughout this section when discussing driver analysis.

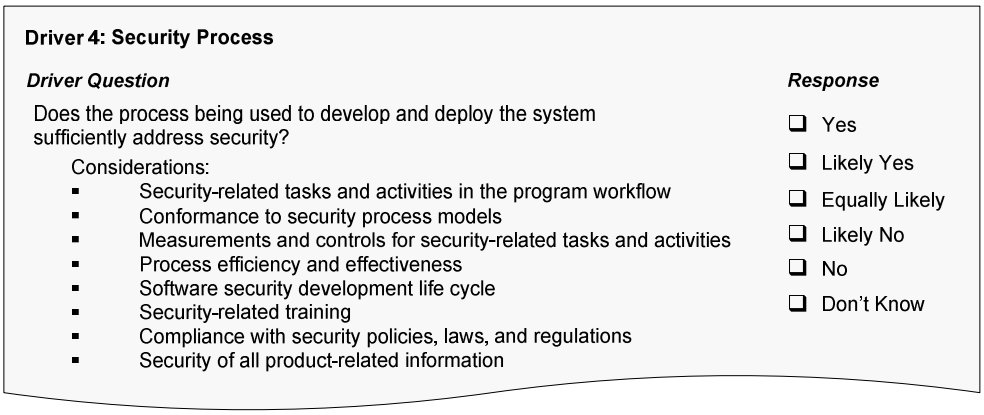

Figure 4: Driver Question and Range of Responses

Because the question in Figure 4 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 standard driver questions for software security along with the considerations for each question are listed in the appendix section of this report.

A set of *driver value criteria*, such as those shown in Figure 5, 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.

The criteria for analyzing a driver must be tailored for each application of driver analysis. For example, the criteria in Figure 5 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.

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 5: Driver Value Criteria

When they analyze a driver, analysts need to consider how conditions and potential events[12] 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*
- 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 6 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 must also be documented because it captures the reasons why analysts selected the response. Any evidence supporting the rationale, such as the results of interviews with system stakeholders and information cited from system documentation must also be cited as well. Recording the rationale and evidence is important for validating the data and associate information products, for historical purposes, and for developing lessons learned.

[12] A *condition* is defined as the current state of being or existence. Conditions define the current set of circumstances that have an impact on system performance. A *potential event* is defined as an occurrence or happening that alters current conditions and, as a result, changes a system's performance characteristics [Alberts 2009].

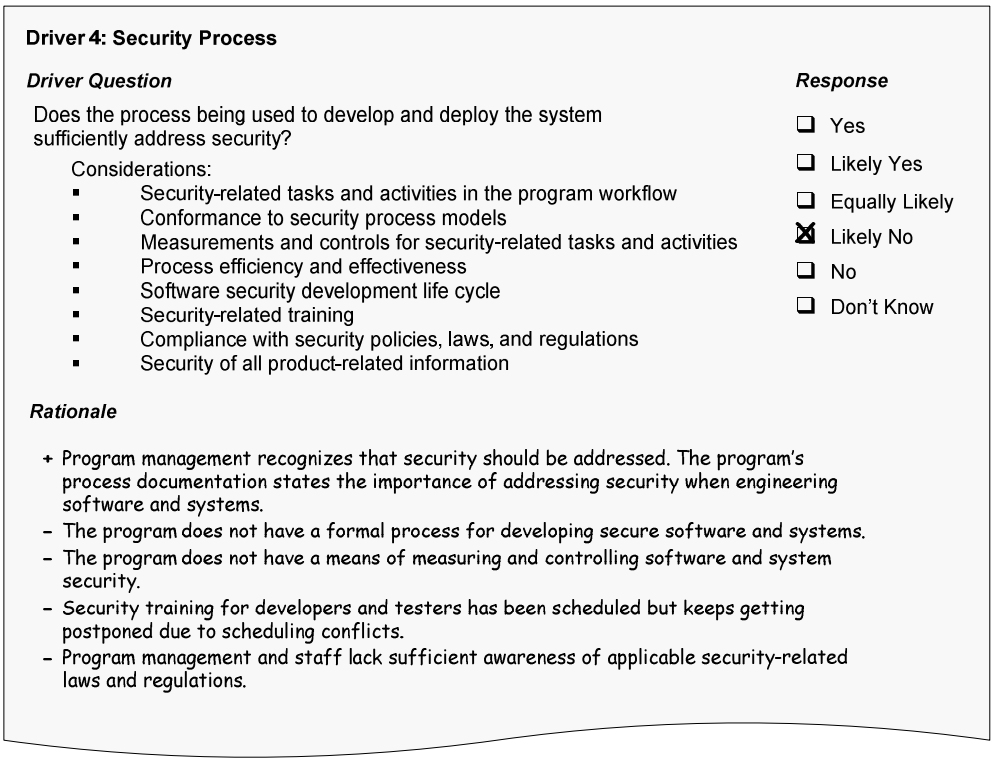

Figure 6: Analyzed Driver

4.3 Driver Profile

A *driver profile* provides a visual summary of the current values 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. It depicts the probability that each driver is in its success state. A high probability for a driver indicates that the driver has a high probability of being in its success state.

Figure 7 provides an example of a driver profile for software security. In Figure 7, a bar graph is used to show 17 drivers that correspond to the standard set for software security, and programmatic drivers are separated from the product drivers. The profile in Figure 7 indicates that the following four drivers have a high probability of being in their failure states: Security Process, Code Security, Integrated System Security, and Product Security Risk Management. The likely states of these four drivers should concern the program's decision makers.

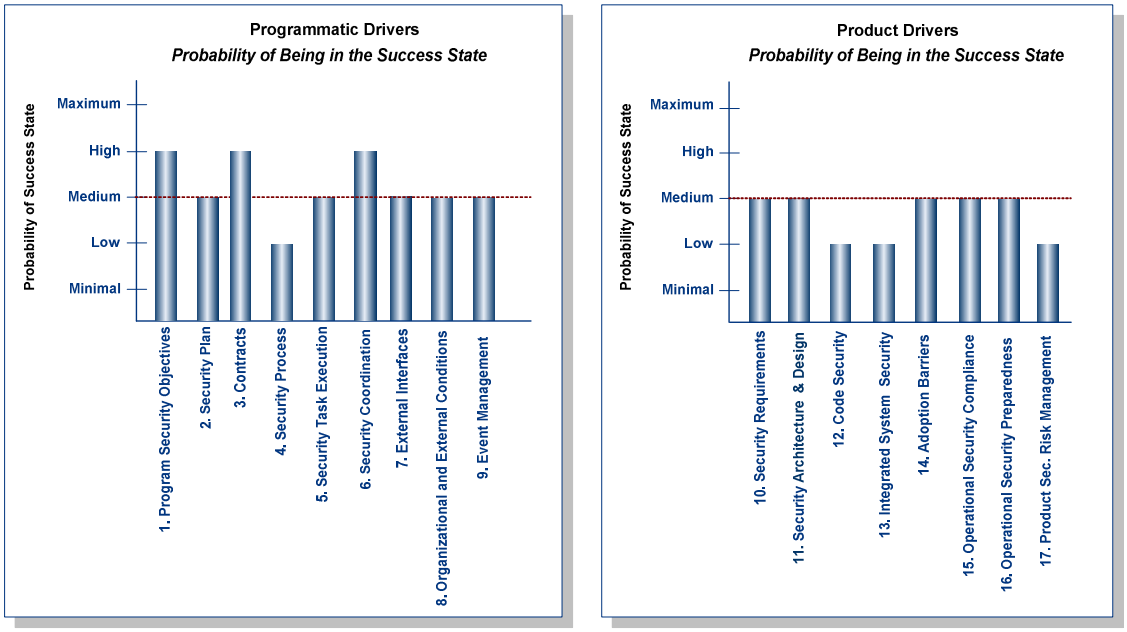

Figure 7: Driver Profile

4.4 Mission Risk

Mission risk is defined as the probability of mission failure (i.e., not achieving key objectives). In this document, the term *mission risk* is used synonymously with the term *systemic risk*. From the MRD perspective *mission risk* is defined as the probability that a driver is in its failure state. As illustrated in Figure 8, a relationship exists between a driver's success state (as depicted in a driver profile) and mission risk.

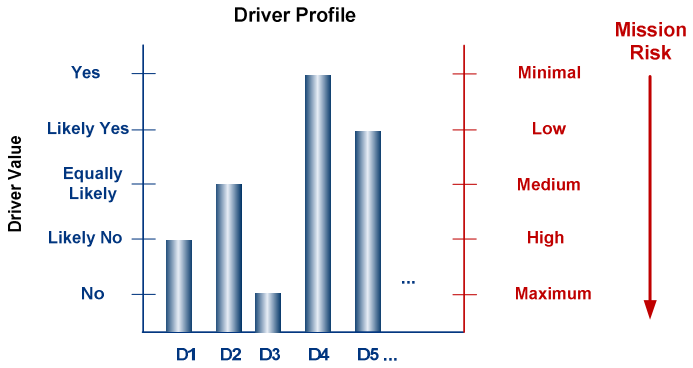

Figure 8: The Relationship between Driver Value and Mission Risk

A driver profile shows the probability that drivers are in their success states. Thus, a driver with a high probability of being in its success state (i.e., a high degree of momentum toward the mission) translates to a low degree of mission risk for that driver. Likewise, a driver with a low probability of being in its success state (i.e., a high probability of being in its failure state) translates to a high degree of mission risk for that driver. The driver profile thus helps decision makers understand how much mission risk is currently affecting a system. Decision makers can then identify actions

intended to increase the probabilities of selected drivers being in their success states and, as a result, mitigate systemic risk to the mission (i.e., mitigate mission risk).

4.5 The MRD: Key Tasks and Steps

Table 3 describes the key tasks and steps that must be performed when conducting the MRD.

Table 3: The MRD: Key Tasks and Steps

Task	Step	Description
Driver Identification	1. Identify the mission	This step establishes the target or focus of the analysis.
	2. Identify the objective(s)	The second step of driver identification determines the tangible outcome(s) that is of interest to decision makers. One or more objectives are identified during this activity.
	3. Identify drivers	Here, analysts establish a small set (typically 10-25) of critical factors that has a strong influence on whether or not the objective(s) will be achieved. These factors are called drivers. At this point, driver identification is complete.
Driver Analysis	4. Evaluate drivers	Once the set of drivers is identified, driver analysis can begin. The first step of driver analysis assesses the value of each driver to determine how it is currently influencing performance.
	5. Document rationale and evidence	This step records the reasons underlying the evaluation of each driver (called the rationale) and any tangible evidence that supports the rationale.
	6. Establish driver profile	The final step of driver analysis produces a visual summary of the current values of all drivers relevant to the mission and objectives being assessed.

The MRD enables systemic risk analysis of interactively complex systems across the life cycle and supply chain. As illustrated throughout this section, the MRD defines an approach for assessing a system's potential for achieving its mission and objectives. Our early work in developing the MRD showed it to be a flexible approach that be applied in many different problems, including software acquisition and development, cyber security, and business portfolio management.

Our current research is focused on applying the MRD in a software security context. The examples provided throughout this section show how we have tailored the approach for software acquisition and development programs. In our current research effort, we are interested in using the MRD to direct an organization's software security measurement and analysis activities. In the next section, we show how the MRD forms the basis for a measurement and analysis framework that integrates software security data from multiple sources.

5 Integrated Measurement and Analysis Framework (IMAF)

The Integrated Measurement and Analysis Framework (IMAF) employs systemic risk analysis to integrate subjective and objective data from a variety of sources, including targeted analysis, status reporting, and measurement, to provide decision makers with a consolidated view of the performance of interactively complex software-reliant systems. We designed the framework for application in a variety of contexts, including acquisition program management, software development, and operational security. However, our long-term research interests are focused on applying the framework in a software security context. In this section, we present the conceptual design of the IMAF from a generic point of view, highlighting its basic structure and key elements. Details about applying the framework in a software security context are deferred to future reports. Figure 9 below illustrates the basic structure of the IMAF.

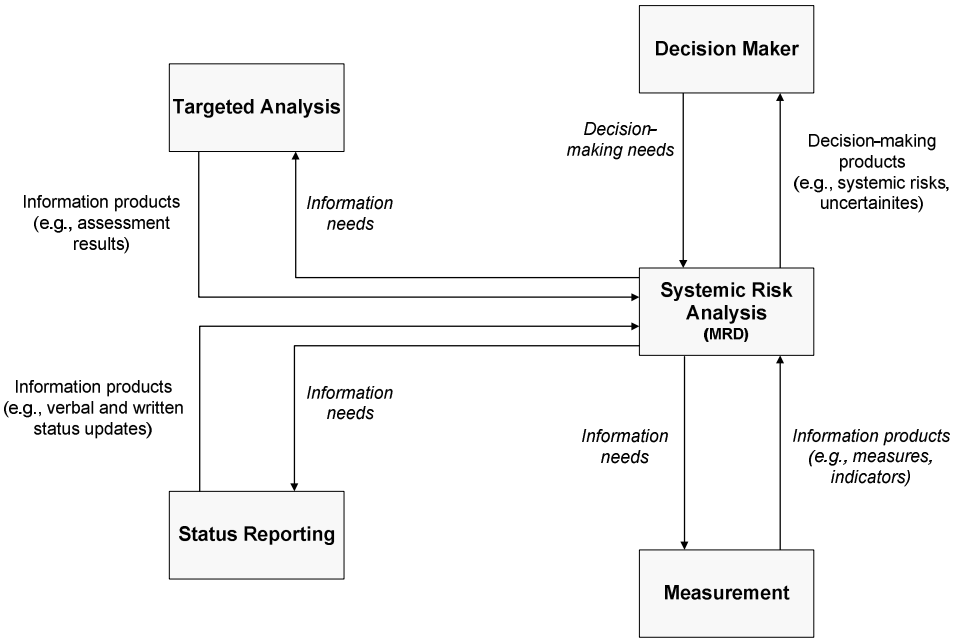

Figure 9: Integrated Measurement and Analysis Framework (IMAF)

The following are the key elements of the IMAF as defined in Figure 9:

- *Decision Maker*—the individual or management team that oversees an interactively complex software-reliant system. The decision maker consumes a variety of information products to satisfy defined decision-making needs.
- *Systemic Risk Analysis*—a risk analysis that examines the aggregate effects of multiple conditions and events on a system's ability to achieve its mission. Systemic risk analysis is conducted to support decision making based on defined information needs and is used within the IMAF to direct measurement, analysis, and reporting activities. The MRD, described in Section 4, provides one way of performing a systemic risk analysis of an interactively complex system.

- *Targeted Analysis*—any analysis that gathers data about specific aspects of components within a system and is conducted to support decision making based on defined information needs. Targeted analysis includes information and knowledge that results from the application of analysis methods, techniques, and tools, such as formal assessments, evaluations, and audits.
- *Status Reporting*—includes verbal, textual, and graphical information products that support defined information needs. Status reports are produced in the form and language that are meaningful for decision makers.
- *Measurement*—activities for selecting, defining, gathering and analyzing measurement data (measures and indicators) based on defined information needs. Measurement data provide decision makers with the quantitative information they need to effectively assess a situation and, as a result, reduce uncertainty.

Measurement, targeted analysis, and status reporting generally provide decision makers with insight into the performance of a system's individual components. However, decision makers often have trouble assessing a system's macro-level behavior from information about its individual components. The IMAF is designed to bridge this gap by integrating performance and quality data for individual components to provide insight into the system's macro-level behavior. It can also highlight where additional data need to be collected based on uncertainties in the integrated data set. For security data, this insight can help identify areas of the system that are vulnerable or are not receiving adequate attention from a security perspective. The next section of this report describes a conceptual scenario of how the IMAF can be used to direct measurement, analysis, and reporting activities and reduce system uncertainty.

5.1 Using the IMAF to Direct Measurement, Analysis, and Reporting Activities

Figure 10 illustrates a scenario that shows how the IMAF can be used to support decision-making activities. The scenario depicted in the figure uses the MRD to direct measurement, analysis, and reporting activities for a given system, such as a software acquisition and development program.

In the scenario, we are making an assumption that measurement, analysis, and reporting data are already being collected on an ongoing basis. This assumption is represented by the first step in the scenario. Most decision makers have a wealth of information at their disposal. Unfortunately, in the internet age information consumers can easily become overwhelmed by too much information. As a result, decision makers can have trouble "connecting the dots" among the disparate types of data that they receive on a daily basis. The IMAF is designed to help decision makers (1) sort through the data they already have, (2) make decisions based on the available data, and (3) determine additional data to collect that will reduce current uncertainties that are present.

In the scenario's second step, a team is chartered to perform the MRD using data that are already being collected. The team conducts the systemic risk analysis and presents the decision maker with the driver profile for the system as well as the following detailed data related to each driver:

- positive conditions that are influencing the driver's state
- negative conditions that are influencing the driver's state
- potential events with positive consequences that *could* influence the driver's state

- potential events with negative consequences that *could* influence the driver's state
- unknown factors that contribute to uncertainty regarding the driver's state
- assumptions that might bias the evaluation of the driver

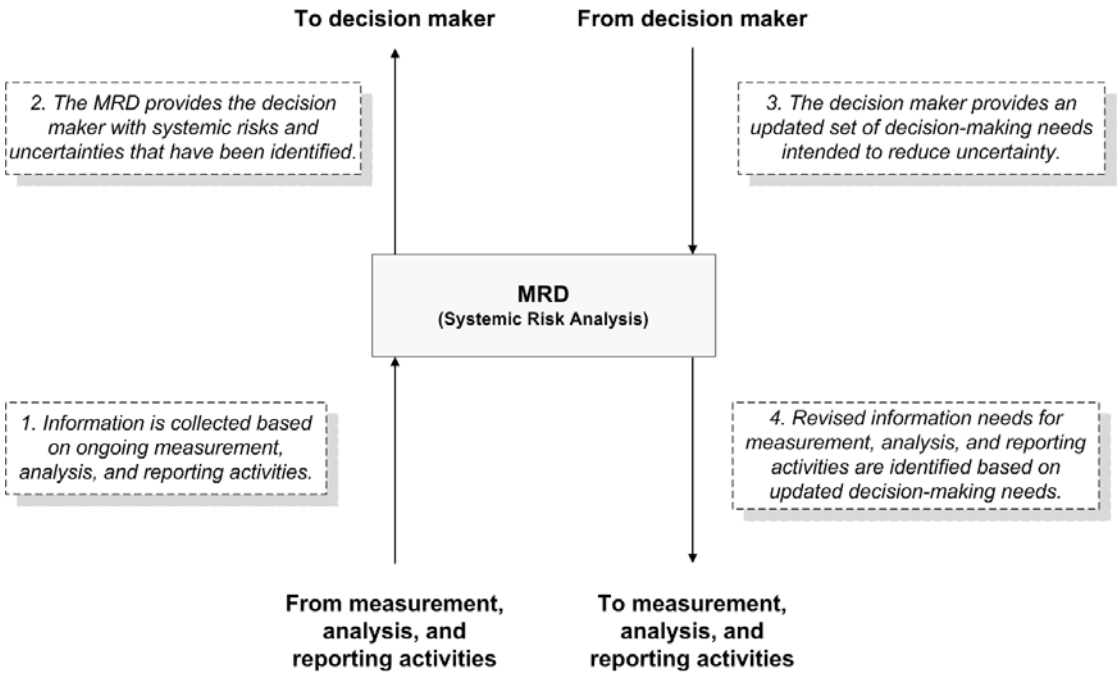

Figure 10: IMAF Scenario

The decision maker typically starts by looking at the driver profile, which establishes a snapshot of systemic risk to the mission (i.e., a snapshot of mission risk). The driver profile enables the decision maker to identify actions intended to increase the probabilities of specific drivers being in their success states, which has the effect of mitigating mission risk.

In addition, the decision maker must look at the uncertainties related to each driver. These uncertainties often reflect circumstances where there are known gaps in the underlying data or where the data collected are not fully trusted. They tend to push a driver's probability toward the middle (i.e., equally likely to be in its success and failure states). Uncertainties provide decision makers an opportunity to collect additional information in order to refine the analysis of a driver.

In the third step of the scenario depicted in Figure 10, the decision maker updates his or her measurement, analysis, and reporting needs/requirements based on the goal of reducing uncertainties related to each driver. Finally, in the fourth step, updated information needs are identified based on the decision maker's revised requirements. These updated information needs can lead to the identification of additional

- assessments to perform
- status information to collect
- measures to collect

The four steps listed in Figure 10 outline a basic process for identifying measurement, analysis, and reporting data that need to be collected. As additional data are collected, the process can be repeated. Over time, the reduction in uncertainty resulting from new data that are collected and analyzed should provide decision makers with more clarity regarding systemic risk to the mission and, as a result, enable better decision making based on more objective data.

5.2 Applying ISO 15939 Measurement in an IMAF Context

The IMAF is a general purpose framework that can be integrated with an organization's measurement, analysis, and reporting practices. Figure 11 illustrates how the ISO 15939 measurement process presented in Section 2 of this report can be applied within an IMAF context. The single measurement box shown in the basic IMAF diagram (Figure 9) has been expanded to include the detailed ISO 15939 measurement process depicted in Figure 2 (in Section 2 of this report). The MRD provides information needs to the *plan measurement* activity of the ISO 15939 measurement process and receives information products, such as measures and indicators, from the *perform measurement* activity.

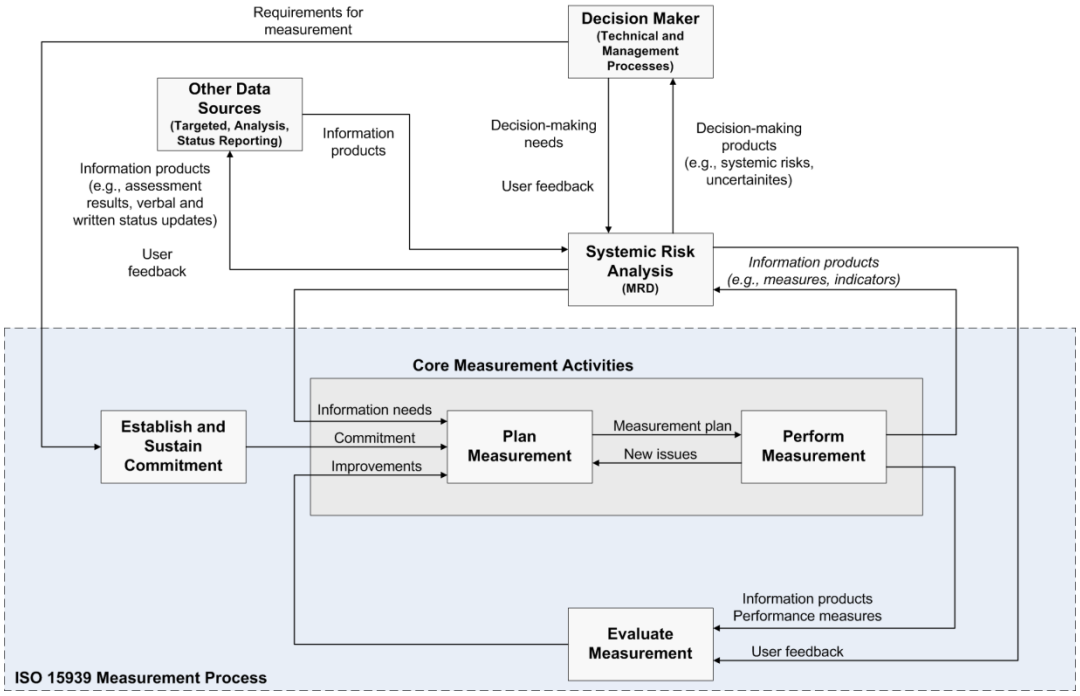

Figure 11: The IMAF in an ISO 15939 Measurement Context

Based on our field work with customers, we believe that the IMAF will help provide decision makers with the information they need, when they need it, and in the right form. The next step in our research and development activities is to begin piloting the framework with customer organizations in a software security context.

6 Additional Research Tasks

The IMAF and the MRD form the foundation for research and development activities being performed by the SSMA project. In this section, we briefly highlight three additional tasks that build on this foundation: (1) measure identification, (2) standard mapping, and (3) driver modeling. Measure identification will enable practitioners to identify and select software security measures based on driver uncertainties (as identified by applying the IMAF). With the standard mapping task, we are developing an approach for linking software security drivers, practices, and measures to the controls specified in commonly used security standards. As part of our driver modeling task, we are beginning to applying predictive analytics in a software security context to enable more informed decision making through quantitative measurement and analysis. Our intent in this section is to provide a conceptual overview of each task. Future reports, white papers, and presentations will provide more in-depth treatments of these three tasks in a software security context.

6.1 Measure Identification

Meaningful measurement and analysis is based on carefully considered and defined measures that are linked to the mission of the system being assessed. Figure 12 provides a conceptual view of how measures can be linked to the mission using the IMAF.

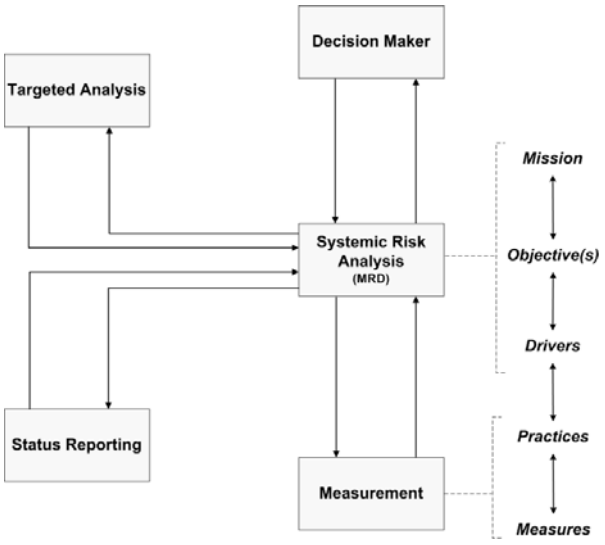

Figure 12: Link from Mission to Measures (Conceptual View)

The discussion of the MRD in Section 4 describes how to decompose a mission into objectives and drivers using driver identification. During driver analysis, the set of drivers is evaluated to determine each driver's influence on the system's mission and objectives. As a result, driver analysis provides decision makers with insight into the degree of systemic risk and uncertainty affecting the mission and objectives.

In previous sections, we conceptually showed how driver uncertainties can be used to define a set of information needs for targeted analysis, status reporting, and measurement. In our measure

identification research task, we are exploring how to derive measures from specified information needs. Given uncertainties related to each driver, we must then answer the following question: How do we then derive meaningful measures that will help reduce driver uncertainties?

To answer this question, we are developing an approach for identifying practices and measures related to a given driver. The approach employs the Goal Question (Indicator) Metric (GQIM) Method developed at the SEI [Park 1996]. The first step in the approach is to determine which practices influence a driver's state. As used in this context, a *practice* is a generally accepted activity (e.g., technique, method, or process) used to achieve a desired goal.

After identifying practices that influence each driver's state, we then derive a set of candidate measures that will provide insight into how practices are implemented as well as how effective they are. Decision makers and analysts can then select which measures from the candidate list will provide the desired reduction in driver uncertainty.

Our work in this area is still early in its development. We have conducted pilot activities where we identified software security practices and candidate software security measures using the standard set of 17 software security drivers (presented in Section 4.1.5 and the appendix). Additional development and piloting activities are needed to test and refine the approach. Further details about this work will be provided in future reports, white papers, and presentations.

6.2 Standard Mapping

As a complement to measure identification, we are also developing an approach for mapping security community standards to software security drivers, practices, and measures. This mapping is conceptually depicted using the dotted, arrowed lines in Figure 13. At the present time, we are mapping the 17 software security drivers along with their associated practices and measures (determined using measure identification) to the controls specified in the NIST 800-53 standard, which is entitled *Recommended Security Controls for Federal Information Systems and Organizations* [NIST 2009].

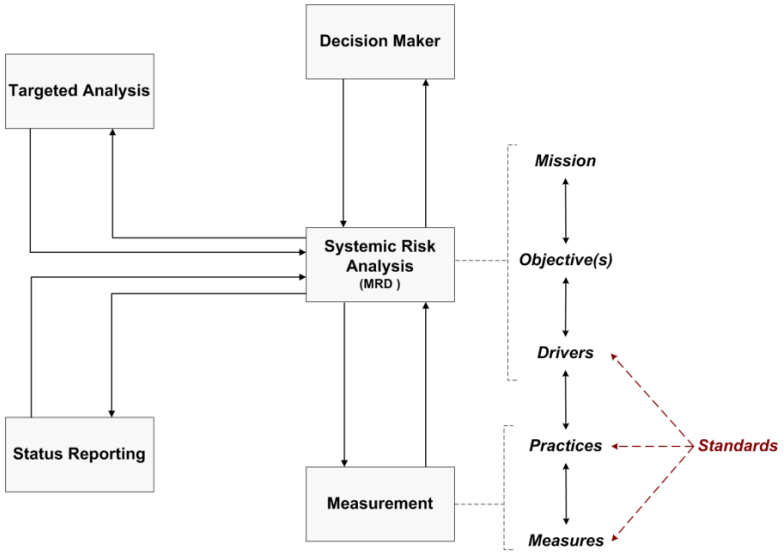

Figure 13: Standard Mapping (Conceptual View)

By mapping security standards to software security drivers, practices, and measures, we can link mission-based measurement and analysis (provided by the IMAF) with an organization's security compliance efforts. Decision makers can identify any conflicts between system performance and the organization's compliance efforts. Our work related to this task is in the prototyping stage, and the early results look promising. However, considerable work still remains. Details about additional development related to standard mappings will be provided in future reports, white papers, and presentations.

6.3 Driver Modeling

While conducting our research and development activities, we identified the need for predictive modeling within the discipline of software security. We came to the conclusion that predictive analytics could provide a basis for quantifying the likelihood of occurrence and relationships among security entities, such as drivers. We also determined that predictive analytics had the potential to enable a more compelling and efficient basis for implementing a measurement and analysis approach prescribed by the IMAF.

We identified a variety of modeling approaches that could be employed to quantitatively implement the IMAF. In particular, we believed that these modeling approaches could provide a predictive analytics engine for the MRD. The candidate approaches that we considered include, but are not limited to

- traditional statistical correlation and regression analysis
- systems dynamics modeling
- Monte Carlo simulation modeling
- probabilistic modeling (e.g., Bayesian Belief Networks)

After considering the demands and constraints affecting our reach project, we selected Bayesian Belief Networks (BBNs) as our modeling approach. Figure 14 shows our initial BBN diagram for the 17 software security drivers that we introduced in Section 4.1.5 of this report.

The BBN in the figure will quantitatively confirm the likelihood of occurrence of each driver's state as well as confirm the relationships of "leading indicators" among the drivers. For example, in Figure 14 each of the drivers, represented by the circled nodes, have one or more states. These could be binary states, such as success and failure, or they could use a scale of 1–5. Additionally, each arrow represents a potential cause-and-effect relationship, or leading indicator relationship. For example, the following five security drivers directly influence the status of the security objective, which is represented by the black circled node in Figure 14:

- driver 13, Integrated System Security
- driver 14, Adoption Barriers
- driver 15, Operational Security Compliance
- driver 16, Operational Security Preparedness
- driver 17, Product Security Risk Management

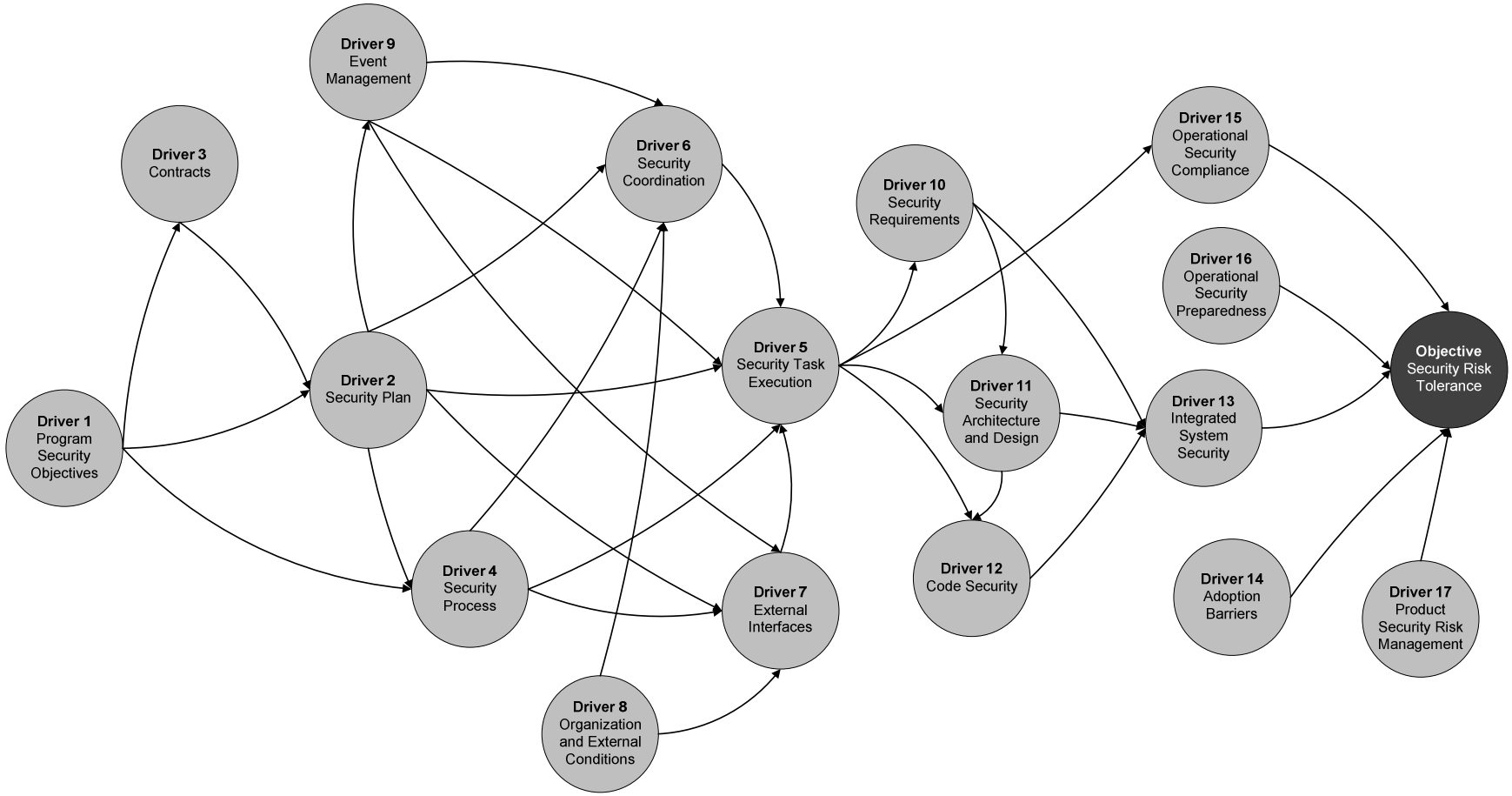

Figure 14 Driver Model (Bayesian Belief Network)

Likewise, the status of driver 11 (Security Architecture and Design) may be predicted with knowledge of driver 5 (Security Task Execution) and driver 10 (Security Requirements).

Figure 14 reflects subjective expert opinion regarding the relationships among the drivers. Over time, empirical analysis of the BBN might demonstrate that some relationships based on expert opinion are not significant, while new relationships might also be identified. The results of this empirical analysis might cause the BBN to be modified, where insignificant relationships are removed from the model and newly discovered relationships are added.

Overall, we believe that an operational BBN model that learns from additional experience and data will prove useful for identifying which drivers have the greatest influence on achieving the security objective. As analysts acquire additional objective or subjective security data about drivers and their relationships, the model will learn from this new information. Probabilities associated with drivers will be updated accordingly, and relationships among drivers in the model will also be updated.

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. Additional details about this work will be provided in future reports, white papers, and presentations.

7 Summary and Next Steps

For several years, the software engineering community has been working to identify practices aimed at developing more secure software. Although some foundational work has been performed throughout the community, efforts to measure software security assurance have yet to materialize in any substantive fashion. As a result, decision makers (e.g., development program and project managers, acquisition program offices) lack confidence in the security characteristics of their software-reliant systems.

In September 2009, the SEI CERT® Program chartered the SSMA Project to advance the state-of-the-practice in software security measurement and analysis. The SSMA Project is researching and developing frameworks, methods, and tools for measuring and monitoring the security characteristics of interactively complex software-reliant systems across the life cycle and supply chain.

The SSMA Project builds on the CERT Program's core competence in software and information security as well as the SEI's work in software engineering measurement and analysis. The main purpose of this project is to address the following two questions:

1. How do we establish, specify, and measure justified confidence that interactively complex software-reliant systems are sufficiently secure to meet operational needs?
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?

This report primarily focuses on answering the first research question. It presents a risk-based approach for establishing, specifying, and measuring justified confidence that interactively complex software-reliant systems are sufficiently secure to meet operational needs.

7.1 The IMAF and the MRD

The main conceptual framework developed under the SSMA project is the Integrated Measurement and Analysis Framework (IMAF), which is depicted in Figure 15. The IMAF employs systemic risk analysis to integrate subjective and objective data from a variety of sources, including targeted analysis, status reporting, and measurement, to provide decision makers with a consolidated view of the performance of interactively complex software-reliant systems.

In general, targeted analysis, status reporting, and measurement activities provide very detailed data about a system's critical components. For interactively complex systems, decision makers often have trouble "connecting the dots" among the very detailed, disparate data available to them. As a result, decision makers can find it difficult to understand a system's macro-level behavior based on available information. The IMAF is designed to bridge this gap by integrating performance data for individual components to provide insight into the system's behavior. It can also highlight where additional data need to be collected based on uncertainties in the integrated data set.

The centerpiece of the IMAF is a systemic risk analysis approach that examines the aggregate effects of multiple conditions and events on a system's ability to achieve its mission. Systemic risk analysis is conducted to support decision making based on defined information needs and is used within the IMAF to direct measurement, analysis, and reporting activities. The SSMA project is developing the Mission Risk Diagnostic (MRD) to enable systemic analysis as prescribed by the IMAF.

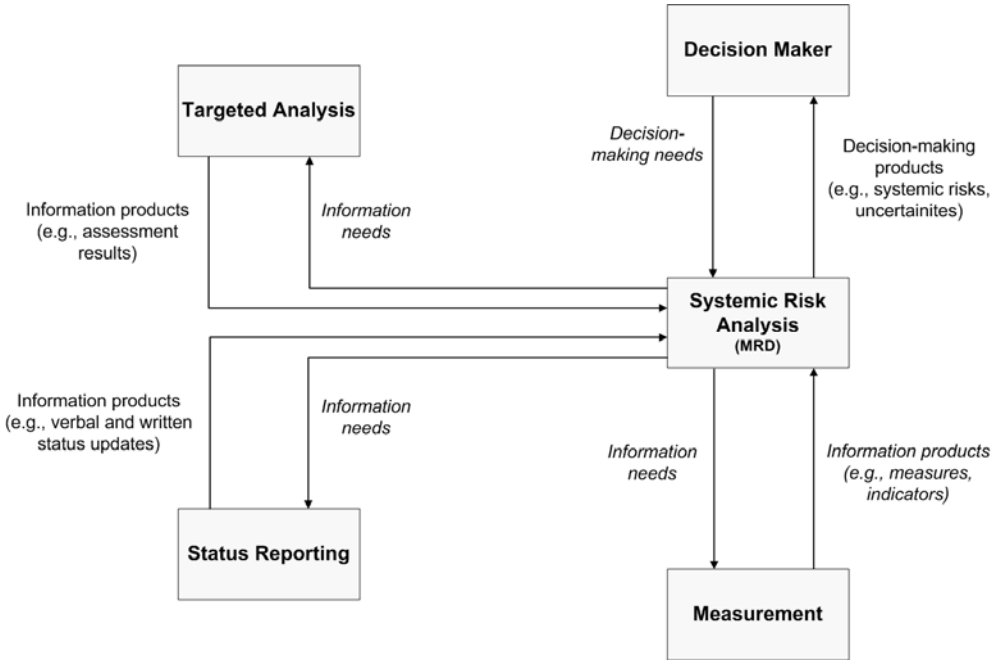

Figure 15: The IMAF Revisited

The MRD comprises two main tasks: driver identification and driver analysis. The main goal of driver identification is to establish 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 established, analysts then employ driver analysis to evaluate each driver in the set.

Driver analysis enables analysts to evaluate the current state of each driver (i.e., how it is affecting current performance) and establish a driver profile for the mission. The purpose of the driver profile is to establish a snapshot of the degree of systemic risk currently affecting the mission (i.e., a snapshot of mission risk). The driver profile enables the decision maker to identify actions intended to increase the probabilities of specific drivers being in their success states, which, in turn, mitigates mission risk.

The decision maker must also consider uncertainties related to each driver. These uncertainties often reflect circumstances where there are known gaps in the underlying data or where the data collected are not fully trusted. They tend to influence a driver's probability toward the middle (i.e., equally likely to be in its success and failure states). Uncertainties provide decision makers an opportunity to collect additional information (via targeted analysis, status reporting, and measurement) in order to refine the analysis of a driver. Over time, the reduction in uncertainty resulting from new data that are collected and analyzed should provide decision makers with more

clarity regarding system performance and, as a result, enable better decision making based on more objective data.

Early versions of the MRD have been piloted in a variety of areas, including software acquisition and development programs, cyber security processes, and business portfolio management. We are currently looking to pilot the IMAF and the MRD in a software security context. The goal is to assess software security during a system's acquisition and development and help decision makers identify software security measures that will help them reduce systemic risk and uncertainty.

7.2 Additional Research

The IMAF and the MRD serve as the foundation for SSMA research and development activities. Building upon this foundation, we have pursued the following three additional research and development tasks during the past two years:

- *measure identification*—an approach for identifying practices and measures related to a given driver
- *standard mapping*—a means of mapping community standards to drivers, practices, and measures
- *driver modeling*— an approach for using predictive analytics as a quantitative basis for implementing the IMAF

Although each of the above tasks is early in its development, early results look promising.

7.3 Next Steps

This report concludes our initial phase of research and development related to software security measurement and analysis. We have established a basis for future measurement and analysis activities though our work in the following areas:

- definition of a measurement and analysis framework (the IMAF)
- development of a method for performing systemic analysis of interactively complex systems (the MRD)
- identifying meaningful measures (measure identification)
- mapping standards to drivers, practices, and measures (standard mapping)
- applying predictive analytics to software security using BBNs (driver modeling)

The main emphasis of our early research and development activities has been the development of the IMAF and the MRD, which have been presented in this report. The goals of our next phase are to (1) pilot and refine the IMAF and the MRD in a software security context and (2) continue research and development activities related to measure identification, standard mapping, and driver modeling. We believe that our work in software security measurement and analysis holds considerable promise for the future. We hope to build on the foundational work described in this report in the years to come.

Appendix: Standard Set of Drivers for Software Security

This appendix provides a prototype set of driver questions for assessing software security as described in Section 4.1.5. 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 address 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**: Does the code sufficiently address security?
13. **Integrated System Security**: Does the integrated system sufficiently address security?

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, standards, and regulations?
16. **Operational Security Preparedness**: Are people prepared to maintain the system's security over time?
17. **Product Security Risk Management**: Is the approach for managing product security risk sufficient?

Driver 1: Program Security Objectives

Are the program's security objectives realistic and achievable?

Considerations:

1. Tradeoffs between security, performance, and other objectives
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
8. Compliance with security policies, laws, standards, and regulations
9. Schedule, funding and, resources allocated to certification and accreditation activities

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
9. Compliance with security policies, laws, standards, and regulations
10. Schedule, funding and, resources allocated to certification and accreditation activities

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 non-compliance issues
7. Compliance with security policies, laws, standards, and regulations

Driver 4: Security Process

Does the process being used to develop and deploy the system sufficiently address 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
8. Security of all product-related information

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

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 commercial off-the-shelf software (COTS), government off-the-shelf (GOTS), and open-source 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
11. Relevant policies, standards, guidelines, and regulations

Driver 12: Code Security

Does the code sufficiently address security?

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. Code interfaces and dependencies as they affect security
9. Process for secure coding
10. Analysis of security threats as they affect code (using methods such as misuse/abuse cases, threat models, and attack patterns)
11. Analysis of common weaknesses
12. Complexity of code
13. Results of risk analysis of the secure code
14. Security of legacy, COTS, GOTS, and open-source software

Driver 13: Integrated System Security

Does the integrated system sufficiently address security?

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, GOTS, and open-source 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. Effect on user productivity
5. Effect on system usability
6. Training provided to users, operators, and maintainers
7. Support provided to users, operators, and maintainers

Driver 15: Operational Security Compliance

Will the system comply with applicable security policies, laws, standards, 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, GOTS, and open-source software
4. Supplier support for legacy systems
5. Degree of anticipated changes to the system
6. Unanticipated use of the system
7. Documented operational policies, practices, and procedures for maintaining operational systems over time
8. Operational staffing levels
9. Experience and expertise of operators, maintainers, customers, 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. Use of operational security tools and techniques (intrusion detection systems, filters, etc.) needed to maintain system security
15. Disaster recovery, contingency and business continuity plans

Driver 17: Product Security Risk Management

Is the approach for managing product security risk sufficient?

Considerations:

1. Process for managing product security risks (identify, analyze, mitigate)
2. Articulation of what constitutes an acceptable (or unacceptable) level of risk
3. Training in the product security risk management approach
4. Compliance with security policies, laws, standards, and regulations
5. Product certification activities
6. Focus on managing product security risks across the life cycle (including requirements, architecture and design, coding, integration, testing, deployment)
7. Consideration of operational (i.e., post deployment) security policies, practices, and procedures

Glossary

condition
the current state of being or existence; conditions define the current set of circumstances that have an impact on system performance

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

driver analysis
a task of the Mission Risk Diagnostic that establishes a set of factors, called drivers, that can be used to measure performance in relation to a program's mission and objectives

driver identification
a task of the Mission Risk Diagnostic that determines how each driver is influencing the objectives

driver profile
a visual summary of the current values of all drivers relevant to the mission and objectives being assessed

interactive complexity
the presence of unplanned and unexpected sequences of events in a system that are either not visible or not immediately understood

interactively complex system
a system whose components interact in relatively unconstrained ways

measure
a variable to which a value is assigned as the result of measurement; a generally accepted technique, method, or process used to complete a task

measurement
a set of observations that reduce uncertainty where the result is expressed as a quantity

mission
the fundamental purpose of the system that is being examined

mission risk
the probability of mission failure (i.e., not achieving key objectives); the probability that a driver is in its failure state

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

potential event
an occurrence or happening that alters current conditions and, as a result, changes a system's performance characteristics

practice
a generally accepted activity (e.g., technique, method, or process) used to achieve a desired goal

product driver
a driver that provides insight into the product that is being acquired, developed, and deployed

program
a group of related projects managed in a coordinated way to obtain benefits and control not available from managing them individually; programs usually include an element of ongoing activity

programmatic driver
a driver that provides insight into how well a system (e.g. an acquisition program) is being managed

project
a planned set of interrelated tasks to be executed over a fixed period of time and within certain cost and other limitations

risk
the probability of suffering harm or loss

socio-technical system
interrelated technical and social elements (e.g., people who are organized in teams or departments, technologies on which people rely) that are engaged in goal-oriented behavior

software security assurance
justified confidence that software-reliant systems are adequately planned, acquired, built, and fielded with sufficient security to meet operational needs, even in the presence of attacks, failures, accidents, and unexpected events

software-reliant system
a socio-technical system whose behavior (e.g., functionality, performance, safety, security, interoperability, and so forth) is dependent on software in some significant way

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

systemic risk
the probability of mission failure (i.e., not achieving key objectives)

systemic risk analysis
a risk analysis that examines the aggregate effects of multiple conditions and events on a system's ability to achieve its mission

tactical risk
the probability that an event will lead to a negative consequence or loss

tactical risk analysis
a risk analysis (based on the principle of system decomposition and component analysis) that evaluates a system's components for potential failures

References

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

[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

[Alberts 2010]
Alberts, Christopher; Allen, Julia; & Stoddard, Robert. *Integrated Measurement and Analysis Framework for Software Security* (CMU/SEI-2010-TN-025), Software Engineering Institute, Carnegie Mellon University, 2010. http://www.sei.cmu.edu/library/abstracts/reports/10tn025.cfm

[Bergey 2009]
Bergey, John K. *A Proactive Means for Incorporating a Software Architecture Evaluation in a DoD System Acquisition* (CMU/SEI-2009-TN-004). Software Engineering Institute, Carnegie Mellon University, 2009. http://www.sei.cmu.edu/library/abstracts/reports/09tn004.cfm

[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

[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

[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.

[NIST 2009]
National Institute of Standards and Technology. *Recommended Security Controls for Federal Information Systems and Organizations,* (NIST Special Publication 800-53). U.S. Department of Commerce, 2009. http://csrc.nist.gov/publications/nistpubs/800-53-Rev3/sp800-53-rev3-final_updated-errata_05-01-2010.pdf

[Park 1996]
Park, Robert; Goethert, Wolfhart; & Florac, William. *Goal-Driven Software Measurement —A Guidebook* (CMU/SEI-96-HB-002). Software Engineering Institute, Carnegie Mellon University, 1996. http://www.sei.cmu.edu/library/abstracts/reports/96hb002.cfm

[Perrow 1999]
Perrow, Charles. *Normal Accidents: Living with High-Risk Technologies*. Princeton University Press, 1999.

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

[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

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

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	February 2012	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Risk-Based Measurement and Analysis: Application to 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-2012-TN-004

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)

For several years, the software engineering community has been working to identify practices aimed at developing more secure software. Although some foundational work has been performed, efforts to measure software security assurance have yet to materialize in any substantive fashion. As a result, decision makers (e.g., development program and project managers, acquisition program offices) lack confidence in the security characteristics of their software-reliant systems. The CERT® Program at Carnegie Mellon University's Software Engineering Institute (SEI) has chartered the Software Security Measurement and Analysis (SSMA) Project to advance the state-of-the-practice in software security measurement and analysis. The SSMA Project is exploring how to use risk analysis to direct an organization's software security measurement and analysis efforts. The overarching goal is to develop a risk-based approach for measuring and monitoring the security characteristics of interactively complex software-reliant systems across the life cycle and supply chain. To accomplish this goal, the project team has developed the SEI Integrated Measurement and Analysis Framework (IMAF) and refined the SEI Mission Risk Diagnostic (MRD). This report is an update to the technical note, *Integrated Measurement and Analysis Framework for Software Security* (CMU/SEI-2010-TN-025), published in September 2010. This report presents the foundational concepts of a risk-based approach for software security measurement and analysis and provides an overview of the IMAF and the MRD.

14. SUBJECT TERMS	15. NUMBER OF PAGES
software security, measurement, measure, distributed environment, systemic analysis, Bayesian Belief Network	63

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