

The slide features a dark blue background with a pattern of lighter blue, overlapping, horizontal arrow-like shapes on the left side. The text is white and positioned on the right side of the slide.

Security Measurement and Analysis

Christopher Alberts
Julia Allen
Robert Stoddard

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

 **Software Engineering Institute** | **CarnegieMellon** © 2011 Carnegie Mellon University

This presentation is entitled “Security Measurement and Analysis.” It describes work being performed by the Software Engineering Institute in the area of security measurement and analysis.

Security Measurement and Analysis

© 2011 Carnegie Mellon University

NO WARRANTY

THIS MATERIAL OF CARNEGIE MELLON UNIVERSITY AND ITS SOFTWARE ENGINEERING INSTITUTE IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This presentation may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

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

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



Software Engineering Institute

CarnegieMellon

Security Measurement and Analysis
© 2011 Carnegie Mellon University

2



Software Engineering Institute | CarnegieMellon

©2011 Carnegie Mellon University

2

Topics

Security Measurement and Analysis (SMA) Project
Software Security Assurance
Frameworks and Protocols
Methods and Tools
Summary



The following five topics will be covered in this presentation:

- Security Measurement and Analysis (SMA) Project
- Software Security Assurance
- Frameworks and Protocols
- Methods and Tools
- Summary



The slide features a dark blue background with a pattern of horizontal, overlapping arrow-like shapes on the left side. The title "Security Measurement and Analysis (SMA) Project" is centered in white text. At the bottom, there is a black footer bar containing the Software Engineering Institute logo, the text "Software Engineering Institute | CarnegieMellon", the text "Security Measurement and Analysis © 2011 Carnegie Mellon University", and a small number "4" on the right.

Topic 1: Security Measurement and Analysis (SMA) Project

Security Measurement and Analysis (SMA) Project: *Objective*

To develop frameworks, methods, and tools for measuring and monitoring the security of large-scale, networked systems across the life cycle and supply chain



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 infrastructures.

The CERT[®] Program at Carnegie Mellon University's Software Engineering Institute (SEI) has chartered the Security Measurement and Analysis (SMA) Project to advance the state of the practice in security measurement and analysis.

The objective of the SMA Project is to develop frameworks, methods, and tools for measuring and monitoring the security of large-scale, networked systems across the life cycle and supply chain.

The SMA Project is focused on measuring and monitoring interactively complex socio-technical systems that span multiple organizational entities. Here, a *socio-technical system* is defined as interrelated technical and social elements that are engaged in goal-oriented behavior. Elements of a socio-technical system include the people who are organized in teams or departments to do their work tasks and the technical systems on which people rely when performing work tasks.



Traditional Measurement and Analysis: *System Decomposition and Component Analysis*

Decompose a system into its constituent components.

Prioritize the components.

Analyze the most critical components.



Traditional measurement and analysis approaches are based on the principle of system decomposition and component analysis, where the first step is to decompose a system into its constituent components. Next, the individual components are prioritized, and only the most critical components are analyzed in detail.



System Decomposition and Component Analysis: *Limitations*

Only critical components are analyzed.

- Noncritical components are not examined.
- Interdependencies among components are not addressed.

Nonlinear relationships (e.g., feedback) are not analyzed. Causal relationships are presumed to be

- simple
- direct
- linear

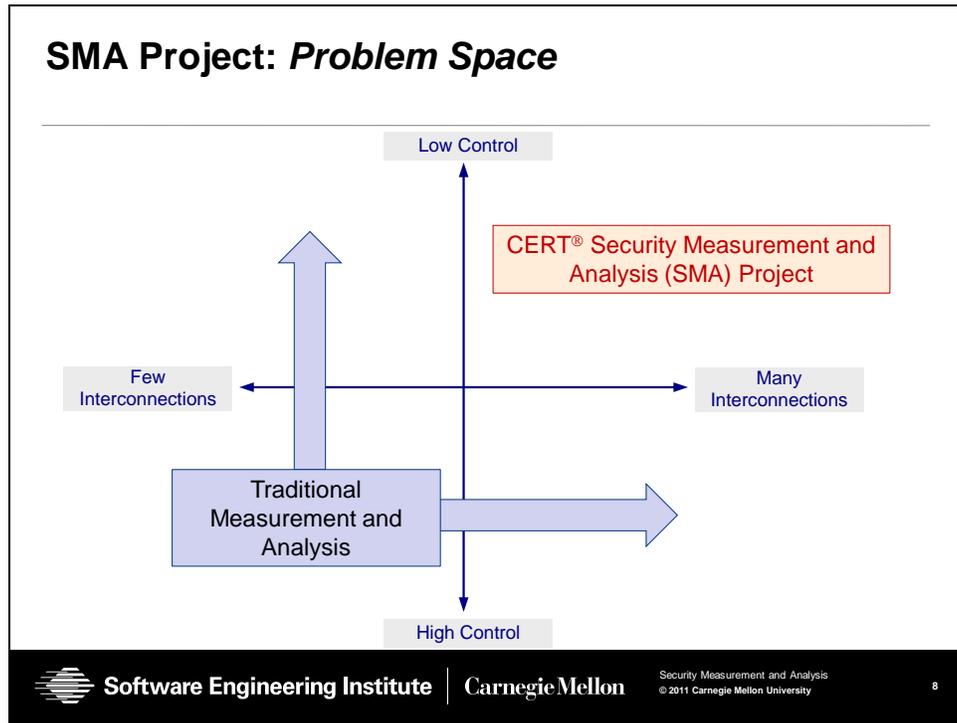
Establishing confidence in the performance of individual components is not sufficient for establishing confidence in the performance of the parent system.



Limitations of system decomposition and component analysis include the following:

- Only critical components are analyzed; noncritical components and interdependencies among components are not addressed.
- Causal relationships are presumed to be simple, direct, and linear. Nonlinear relationships, such as feedback, are not analyzed.
- Confidence in the performance of critical components is not sufficient for establishing confidence in the performance of the parent system (or the parent system of systems).





Note: This slide is a build.

The problem space is defined by the following two dimensions: (1) the degree of management control over a system and (2) the extent to which the system is interconnected.

Traditional measurement and analysis approaches, which employ system decomposition and component analysis, are extremely effective in high-control environments with few interconnections. However, traditional approaches also scale to (1) high-control environments with many interconnections (e.g., using modeling and simulation) and (2) low-control environments with few interconnections (e.g., using collaborative approaches and information sharing among participants).

The SMA Project is focusing on the upper right quadrant in the grid, low-control environments with many interconnections. Traditional measurement and analysis approaches do not readily scale to low-control, highly interconnected environments.

SMA Project: *Distributed Management Environments*

Multiple, independently managed organizational entities working collaboratively to acquire, develop, and operate large-scale, networked systems

- acquisition programs
- development programs
- software supply chains

The diagram is divided into two parts: Programmatic Complexity and Product Complexity. Programmatic Complexity shows three organizations (A, B, and C) with their own internal structures and inter-organizational dependencies. Organization A has components A1, A2, A3, and A4. Organization B has B1, B2, B3, and B4. Organization C has C1, C2, and C3. Arrows indicate dependencies between these components across organizations. Product Complexity shows a networked system with four organizations: Organization C (Data Provider), Organization A (Data User), Organization D (System Maintenance), and Organization E (Data Manager). Each organization has its own internal components and is connected to a central network cloud. Other organizations are also shown connected to the network.

Programmatic Complexity **Product Complexity**

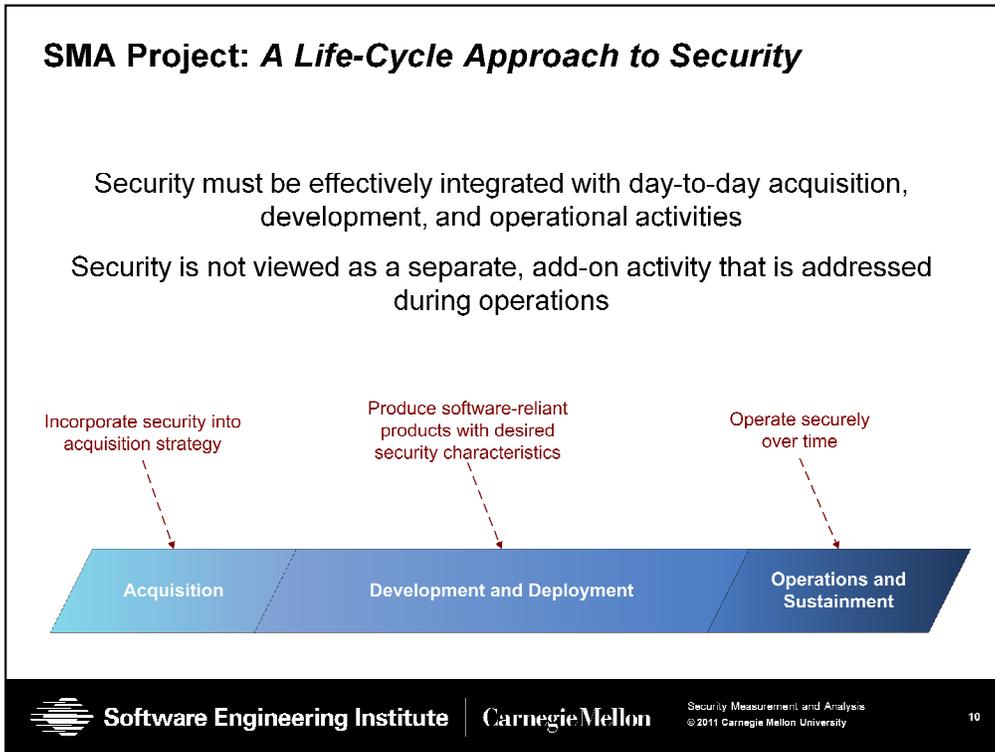
Software Engineering Institute | **CarnegieMellon** Security Measurement and Analysis
© 2011 Carnegie Mellon University 9

A *distributed management environment* is defined as multiple, independently managed organizational entities working collaboratively to achieve a common mission or purpose.

In general, no single administrative structure or set of policies governs all organizations in a distributed management environment such as a software supply chain. In addition, no single manager has authority over all organizations within the environment. Multiple points of management control (i.e., multiple decision makers) exist, which creates a degree of programmatic complexity that can be difficult to manage effectively.

Examples of distributed management environments include large Department of Defense (DoD) acquisition and development programs as well as software supply chains.

Software products produced by distributed management environments tend to comprise many integrated components, which leads to programmatic complexity that can be difficult to manage effectively.



The SMA Project takes a life-cycle approach to addressing security, where security must be effectively integrated with day-to-day acquisition, development, and operational activities. As a result, security is not viewed as a separate, add-on activity that is addressed during operations, which differs from current practice.

SMA Project: *Initial Application Area*

Software security assurance is the project's initial application area.

Future work might address other aspects of security, for example

- incident management
- operational security
- others



The SMA project is initially focused on measuring and monitoring within a software security assurance context. However, SMA frameworks, methods, and tools can be applied in other contexts as well. Future work might address other aspects of security, for example, incident management or operational security.





The slide features a dark blue background with a series of horizontal, overlapping arrow-like shapes on the left side, pointing towards the center. The text "Software Security Assurance" is centered in white. At the bottom, there is a black footer bar containing the Software Engineering Institute logo, the text "Software Engineering Institute | CarnegieMellon", the text "Security Measurement and Analysis © 2011 Carnegie Mellon University", and the page number "12".

Software Security Assurance

 **Software Engineering Institute** | CarnegieMellon

Security Measurement and Analysis
© 2011 Carnegie Mellon University

12

Topic 2: Software Security Assurance

Software Security Assurance: *SEI Definition*

The level of 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
- unexpected events

Software security assurance is focused on the security aspect of software assurance.



A common definition of software assurance is “the level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at any time during its life cycle, and that the software functions in the intended manner.” [1]

An expanded definition of software assurance is “the application of technologies and processes to achieve a required level of confidence that software systems and services function in the intended manner, are free from accidental or intentional vulnerabilities, provide security capabilities appropriate to the threat environment, and recover from intrusions and failures.” [2]

The SMA Project defines *software security assurance* as 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 security assurance is thus focused on the security aspect of software assurance.

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.

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

[2] Mead, N., Allen, J., Ardis, M., Hilburn, T., Kornecki, A., Linger, R., and McDonald, J. *Software Assurance Curriculum Project Volume I: Master of Software Assurance Reference Curriculum* (CMU/SEI-2010-TR-005). Software Engineering Institute, Carnegie Mellon University, 2010. <http://www.sei.cmu.edu/library/abstracts/reports/10tr005.cfm>



Software Security Assurance Activities

Assess

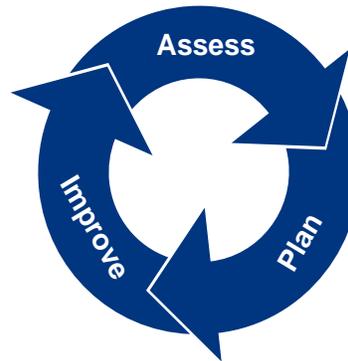
Establish the current level of software security assurance.

Plan

Develop a plan to maintain or improve the current level of software security assurance.

Improve

Take planned action to maintain or improve the current level of software security assurance, and track the plan to completion.



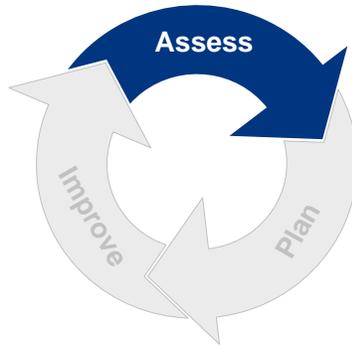
Management of software security assurance comprises the following three activities:

- **Assess** – Establish the current level of software security assurance.
- **Plan** – Develop a plan to maintain or improve the current level of software security assurance.
- **Improve** – Take planned action to maintain or improve the current level of software security assurance, and track the plan to completion.



Role of Assessment in Software Security Assurance

Independent assessment is the vehicle for establishing confidence that large-scale, networked systems will be adequately secure to meet operational needs.



The foundation of a software security assurance capability is the ability to assess assurance effectively. Independent assessment is the vehicle for establishing confidence that large-scale, networked systems will be adequately secure to meet operational needs.



SMA Project: *Products and Services*

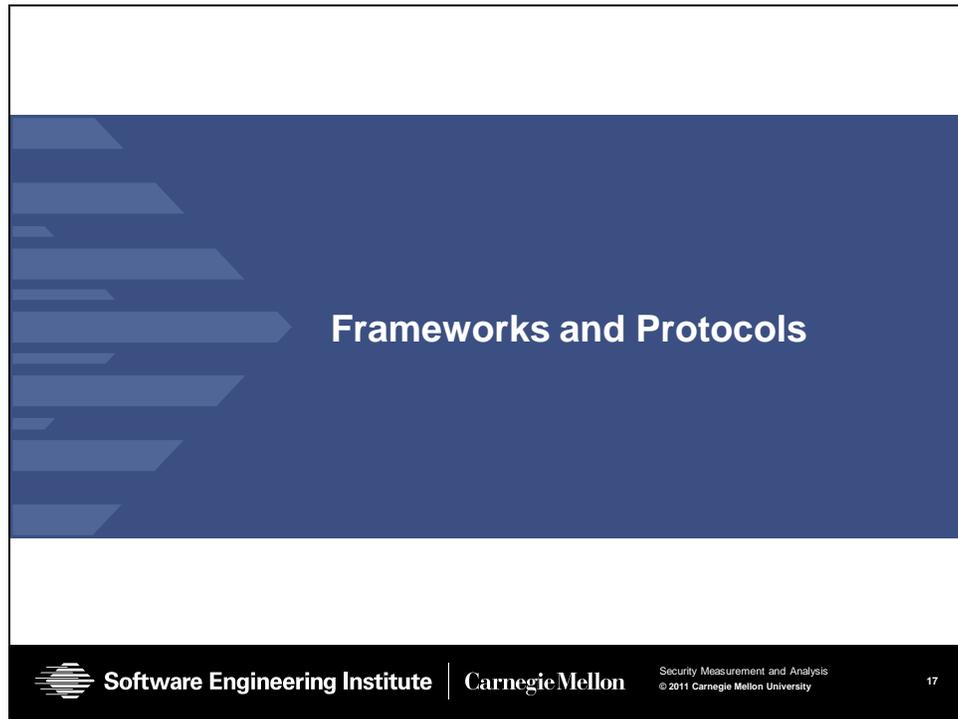
Type	Objective	Products and Services
Frameworks and Protocols	To define the foundational elements of software security assurance measurement and analysis	<ul style="list-style-type: none"> ▪ Integrated Measurement and Analysis Framework (IMAF) ▪ Mission-Objective-Driver (MOD) Protocol ▪ Practice-and-Standard Mappings
Methods and Tools	To provide software security assurance measurement-and-analysis solutions that practitioners can apply	<ul style="list-style-type: none"> ▪ Software Security Review (SSR) ▪ Multi-View Decision Making (MVDM) ▪ Software Security Measurement ▪ Model-Based SSR
Curriculum and Certification	To enable practitioners to apply software security assurance measurement-and-analysis solutions	<ul style="list-style-type: none"> ▪ Software Security Assurance Courses


Software Engineering Institute | **CarnegieMellon**
Security Measurement and Analysis
© 2011 Carnegie Mellon University
16

The SMA Project is developing the following three types of products and services:

- **Frameworks and Protocols** – the foundational elements of software security assurance measurement and analysis
- **Methods and Tools** – software security assurance measurement-and-analysis solutions that practitioners can apply
- **Curriculum and Certification** – courses and certification programs that enable practitioners to apply software security assurance measurement-and-analysis solutions

This presentation examines the first two types, (1) Frameworks and Protocols and (2) Methods and Tools.



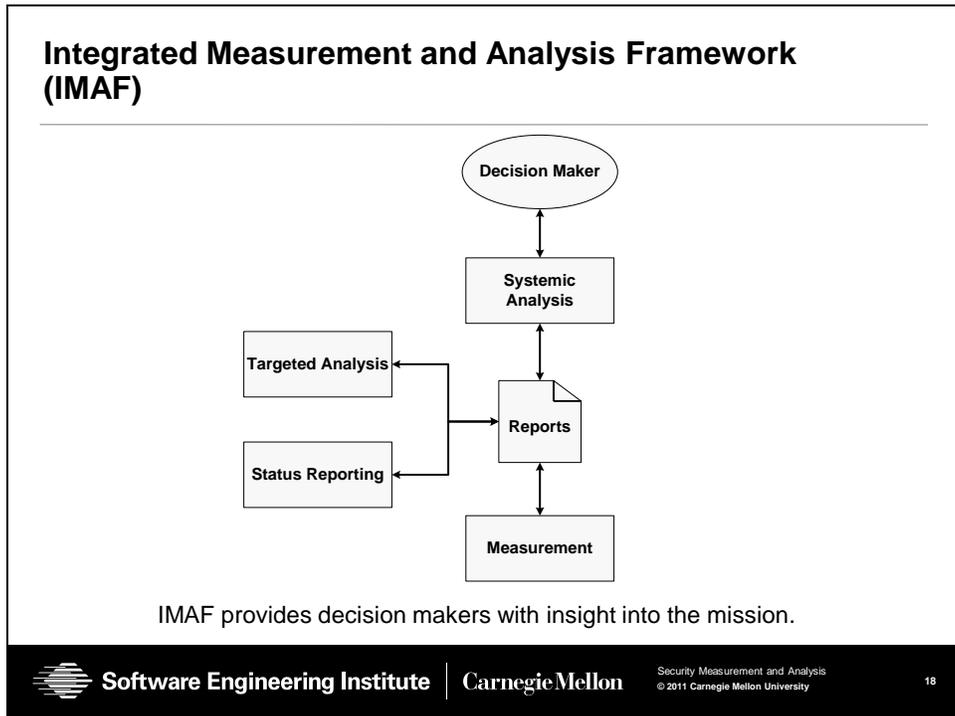
Frameworks and Protocols

Software Engineering Institute | Carnegie Mellon

Security Measurement and Analysis
© 2011 Carnegie Mellon University

17

Topic 3: Frameworks and Protocols



The SMA Project defines a *framework* as a basic conceptual structure that highlights the relationships among a collection of components. A *protocol* is defined as the sequence of activities that must be completed when conducting a method. While a protocol defines what needs to be accomplished, it does not specify how to perform those activities.

The SEI Integrated Measurement and Analysis Framework (IMAF) employs systemic 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 large-scale, networked systems.

Systemic Analysis

Systemic analysis is based on system theory.

The goal is to analyze a system as a whole.

Some system properties are best analyzed by considering the entire system, including

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

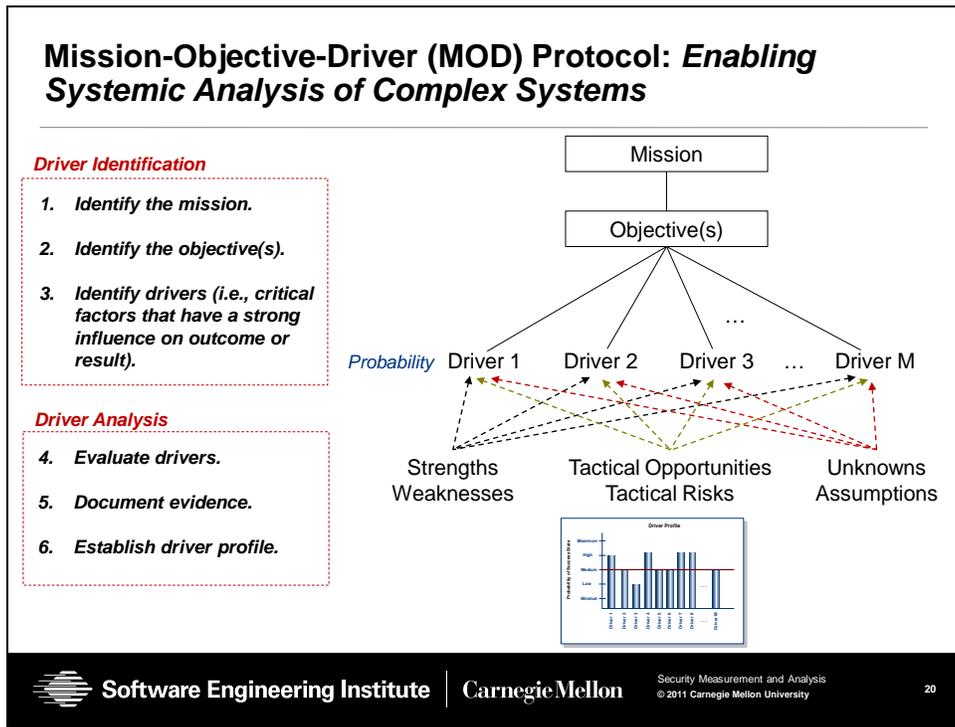


Systemic analysis is based on system theory. The underlying goal of system theory is to analyze a system as a whole rather than decomposing it into individual components and then analyzing each component separately [3]. In fact, some properties of a system are best analyzed by considering the entire system, including

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

[3] Leveson, Nancy. "A New Accident Model for Engineering Safer Systems." *Safety Science* 42, 4 (April 2004): 237-270.





The Mission-Objective-Driver (MOD) Protocol specifies an approach for performing systemic analysis of interactively complex socio-technical systems. The following two activities form the foundation of the MOD Protocol: (1) driver identification and (2) driver analysis.

The main goal of driver identification is to identify a set of factors, called drivers, that can be used to measure performance in relation to a program’s mission and objectives. Refer to slide 32 of this presentation for the standard set of drivers for software security.

Once the set of drivers is identified, analysts can then evaluate each driver in the set to gain insight into the likelihood of achieving the mission and objectives. To measure performance effectively, analysts must ensure that the set of drivers conveys sufficient information about the mission and objectives being evaluated.

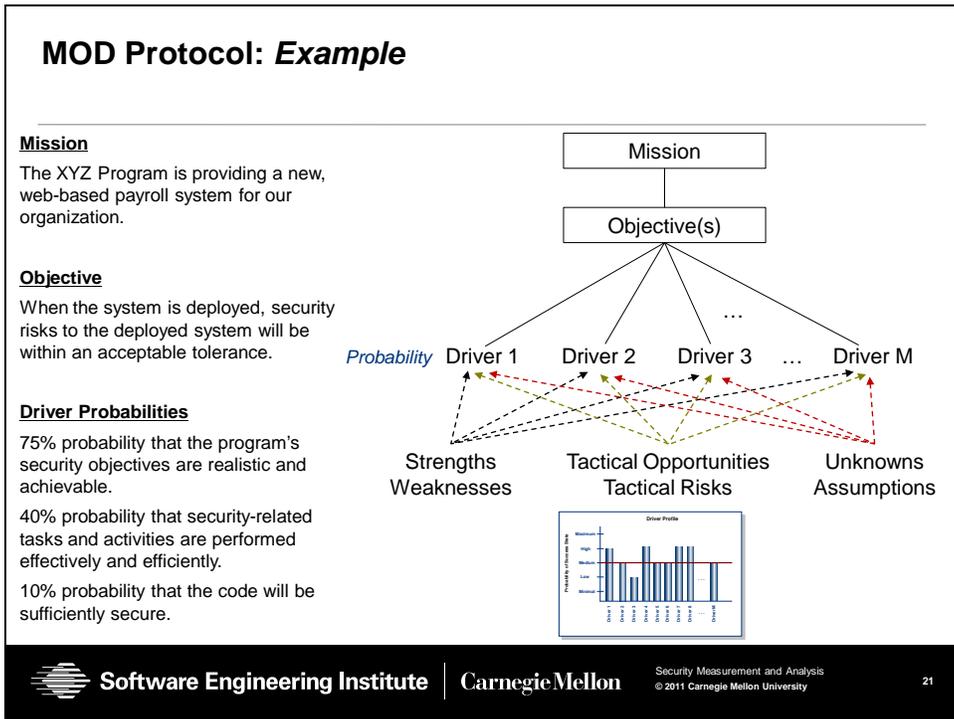
Driver identification comprises the following protocol activities:

- Identify the mission.
- Identify the objective(s).
- Identify drivers (i.e., critical factors that have a strong influence on outcome or result).

The goal of driver analysis is to determine how each driver is influencing the objectives. Analysts must determine whether each driver is guiding the system toward its objectives (success driver) or away from its objectives (failure state).

Driver analysis comprises the following protocol activities:

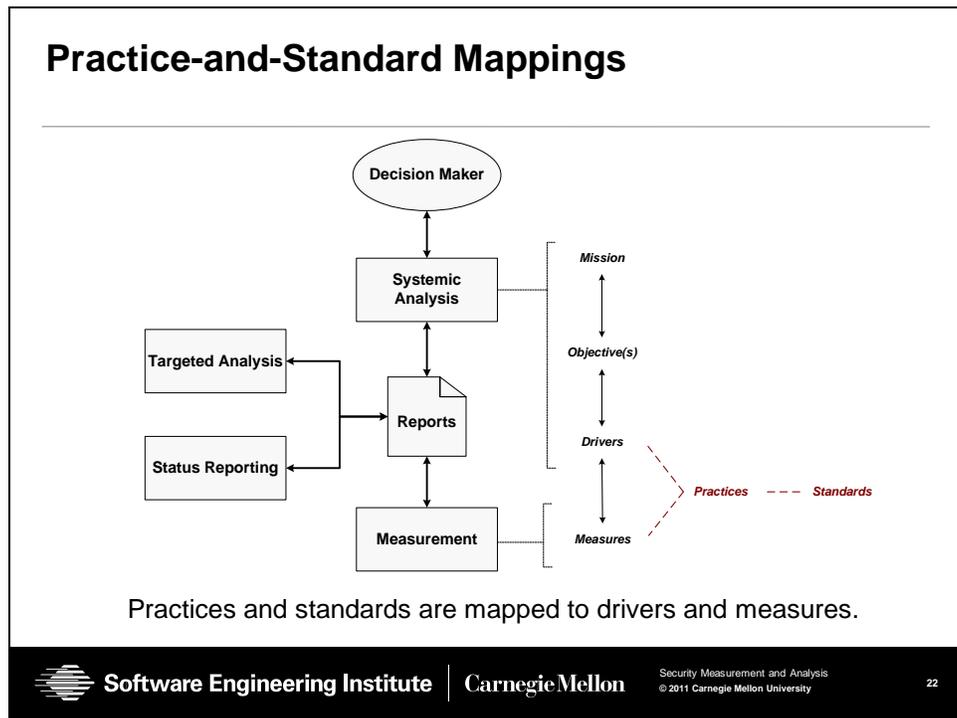
- Evaluate drivers.
- Document evidence.
- Establish driver profile.



The term *mission* is defined as the fundamental purpose of an individual, group, or operation. An example of a mission is: The XYZ Program is providing a new, web-based payroll system for our organization. An *objective* is defined as a tangible outcome or result that must be achieved when pursuing a mission. An example of an objective is: When the system is deployed, security risks to the deployed system will be within an acceptable tolerance.

A *driver* is a factor that has a strong influence on the eventual outcome or result (i.e., whether or not objectives will be achieved). The following are examples of evaluated drivers:

- 75% probability that the program's security objectives are realistic and achievable.
- 40% probability that security-related tasks and activities are performed effectively and efficiently.
- 10% probability that the code will be sufficiently secure.



Performing meaningful measurement and analysis based on carefully considered and defined software security measures requires a clear statement of the mission or purpose. This statement is further expanded into a set of objectives that reflect the mission. A set of drivers can be derived from the objectives to define a set of factors that has a strong influence on the eventual outcome or result (i.e., whether or not objectives will be achieved).

A measurement is an observation that results in information (reduction of uncertainty) about a quantity. [4] A measure is the value assigned to a variable that is used to provide a decision maker with insight into a given characteristic or property of an entity. Measures can be linked to drivers. As shown in the slide, IMAF provides a line of sight from mission to measures.

As illustrated on the slide, drivers and measures can also be mapped to practices and standards. For a given mission and objectives, decision makers can (1) assess confidence in achieving the mission and objectives and (2) gauge performance in relation to practices and standards.

The SMA Project team has begun to develop mappings for the following two standards: NIST 800-53 and ISO 27002.

[4] Hubbard, Douglas. *Applied Information Economics Seminar: Executive Overview*. Hubbard Decision Research, 2010. <http://www.hubbardresearch.com/>

Practice Mapping: *Example*

Driver 10. Security Requirements: Requirements sufficiently address security.

Type	Practice	Measure
Implementation Measure	Product security requirements are documented.	% of software products for which security requirements are/are not documented
Effectiveness Measure	Product security requirements adequately address customer, user, and stakeholder requirements and needs.	% of security requirements that meet (do not meet) customer-, user-, and stakeholder-defined thresholds for adequacy
Process Performance Measures	The process used to specify security requirements performs as expected.	Extent to which the defined process for specifying security requirements meets its performance criteria



This slide presents an example practice mapping for Driver 10, Security Requirements. (This example is not complete.) The following three types of measures are included in this example:

- implementation measure
- effectiveness measure
- process performance measure



Example: NIST 800-53 (a)

Family and Class	Control	Related Controls
SI. System and Information Integrity	SI-2 Flaw Remediation The organization a) identifies, reports, and corrects information system flaws b) tests software updates related to flaw remediation for effectiveness and potential side effects on organizational information systems before installation c) incorporates flaw remediation into the organizational configuration management process	CA-2, CA-7, CM-3, MA-2, IR-4, RA-5, SA-11, SI-11



This slide and the next present an example mapping for NIST 800-53. (This example is not complete.)

This slide presents information from the NIST 800-53 standard related to family and class, control, and related controls.



Example: NIST 800-53 (b)

Guidance	Related Drivers	Practices	Measures
<p>2. The organization (including any contractor to the organization) promptly installs security-relevant software updates (e.g., patches, service packs, and hot fixes).</p> <p>Organizations are encouraged to use resources such as the Common Weakness Enumeration.</p>	<p>16. Operational Security Preparedness</p> <p>7. External Interfaces</p>	<p>Security-relevant software updates are installed for all software components with software flaws and vulnerabilities where corrective action is required.</p> <p>Security-relevant software updates are installed in a timely manner. "Updates" as used here may also include other mitigating actions that do not involve a change to the software.</p>	<ul style="list-style-type: none"> • % of software components requiring security-relevant software updates • % of software components requiring security-relevant software updates where such updates have been installed



This slide continues the example from the previous slide. Here, the first column presents specific guidance for the control that was featured on the previous slide.

The three columns to the right of the Guidance column show our mapping of the guidance to

- related drivers
- practices
- measures

Refer to slide 32 of this presentation for the standard set of drivers for software security.



Example: *ISO 27002 (a)*

Security Clause	Security Topic	Control Objective	Control
12. Information systems acquisition, development, and maintenance	12.1. Security requirements of information systems	To ensure that security is an integral part of information systems	12.1.1 Security requirements analysis and specification Statements of business requirements for new information systems, or enhancements to existing information systems should specify the requirements for security controls.



This slide and the next present an example mapping for ISO 27002. (This example is not complete.)

This slide presents information from the ISO 27002 standard related to security clause, security topic, control objective, and control.



Example: ISO 27002 (b)

Guidance	Related Driver	Practice	Measures
2. Security requirements justified, agreed, and documented as part of the business case for an information system (Objective).	10. Security Requirements	Security requirements are documented as part of the business case	<ul style="list-style-type: none">• % of system components for which security requirements are/are not documented as part of the business case for the information system• % of business cases for information systems that include/do not include security requirements for the system components that reside on the system

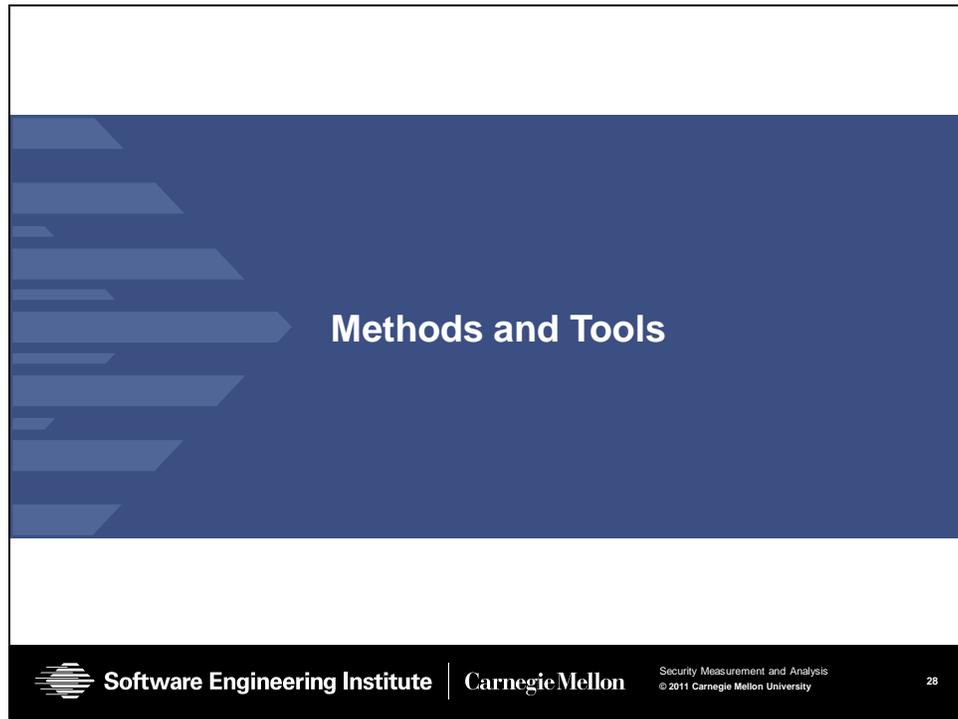


This slide continues the example from the previous slide. Here, the first column presents specific guidance for the control that was featured on the previous slide.

The three columns to the right of the Guidance column show our mapping of the guidance to

- related drivers
- practices
- measures





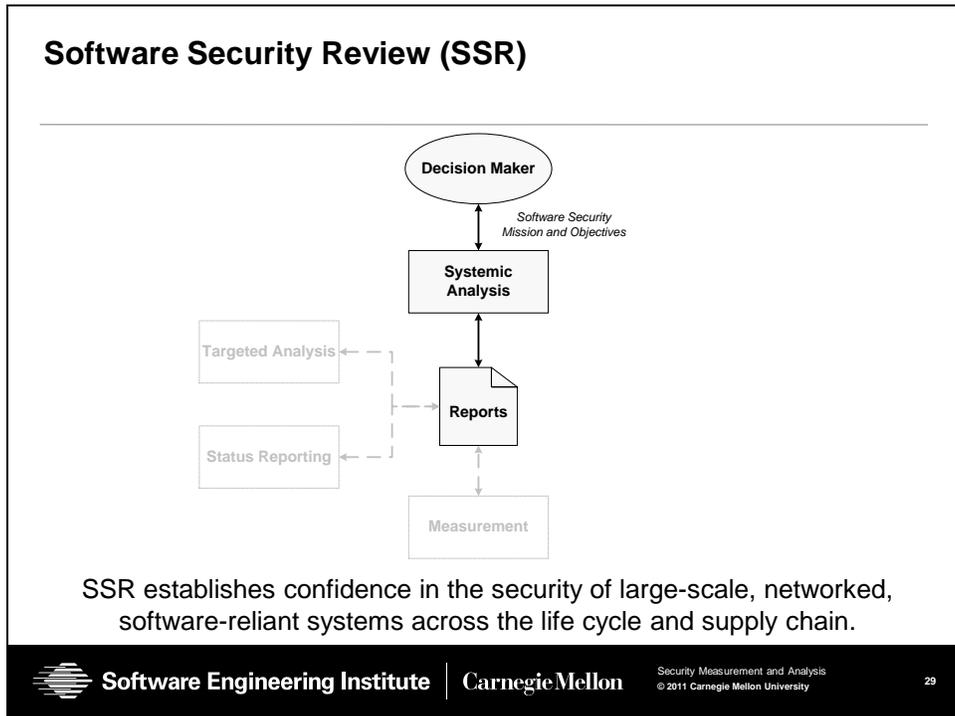
Methods and Tools

Software Engineering Institute | Carnegie Mellon

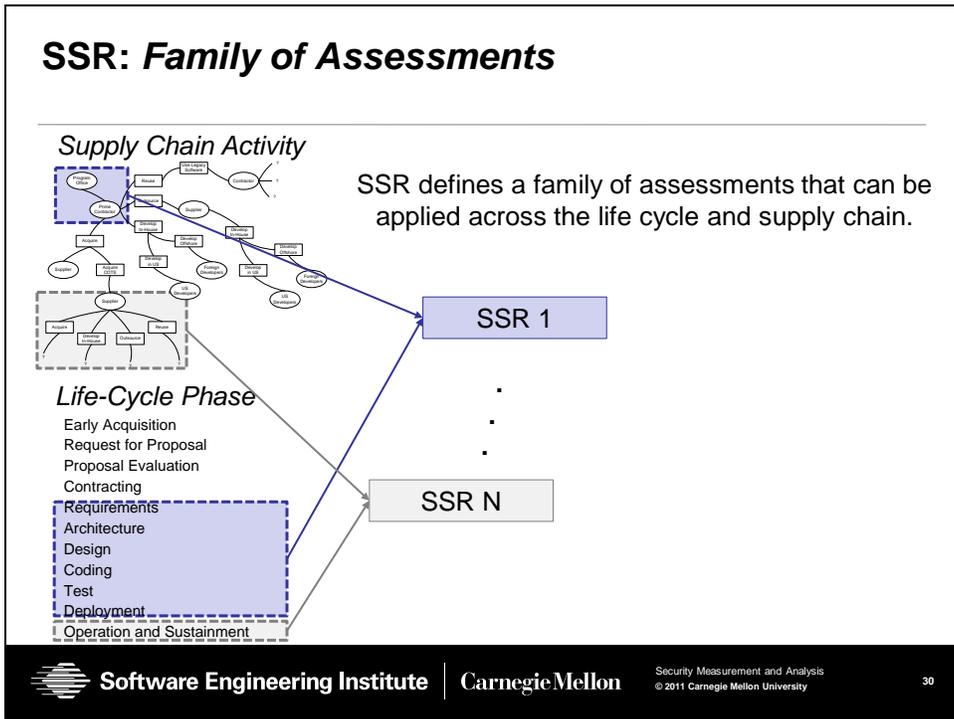
Security Measurement and Analysis
© 2011 Carnegie Mellon University

28

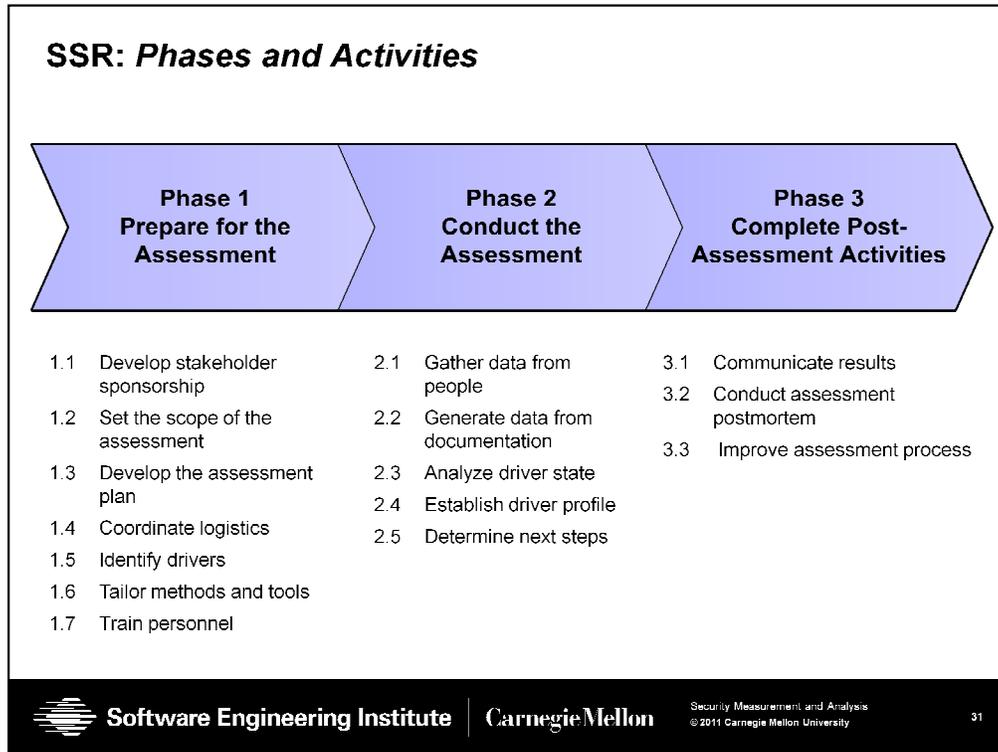
Topic 4: Methods and Tools



The *Software Security Review (SSR)* is a method conducted by independent teams to assess the security characteristics of software-reliant systems. SSR is a driver-based approach that can be used to measure and monitor software security assurance across the life cycle and supply chain (including acquisition, development, and operations).



SSR defines a family of assessments that can be applied across the life cycle and supply chain. An SSR method can be tailored based on life-cycle phase and supply chain activity.



Phase 1 of the SSR Method, *Prepare for the Assessment*, is focused on getting ready to conduct the assessment. This includes all of the planning and logistics management needed to make the assessment execution flow smoothly as well as assuring that key stakeholders provide visible support for the assessment. This preparation lays the foundation for conducting the assessment during Phase 2. Phase 1 comprises seven activities.

During Phase 2, the core assessment activities are performed. During this phase, data are gathered from people and generated from relevant documentation. These data are then used to evaluate a set of key drivers and ultimately construct a driver profile. Decision-makers then determine whether the current state is acceptable and identify actions for maintaining or improving the current state. Phase 2 includes five activities.

Phase 3 conveys the results of an SSR assessment to key stakeholders and identifies actions that can help the efficiency and effectiveness of future SSR assessments. The objective when communicating assessment results to stakeholders is to present findings in a format that meets their needs and requirements. Different formats might be needed to communicate results to different types of stakeholders.

A postmortem is used to identify and document ways in which the SSR assessment can be improved. Updates and improvements to SSR assessment procedures, artifacts, tools, and training are made as appropriate. Postmortems are usually conducted after a given assessment. However, they can also be held on a more periodic basis if multiple assessments are planned. Phase 3 consists of three activities.

SSR: Drivers for Secure Software Development (Draft)

Objectives

1. Program Security Objectives

Preparation

2. Security Plan
3. Contracts
4. Security Process

Execution

5. Security Task Execution
6. Security Coordination
7. External Interfaces

Environment

8. Organizational and External Conditions

Resilience

9. Event Management

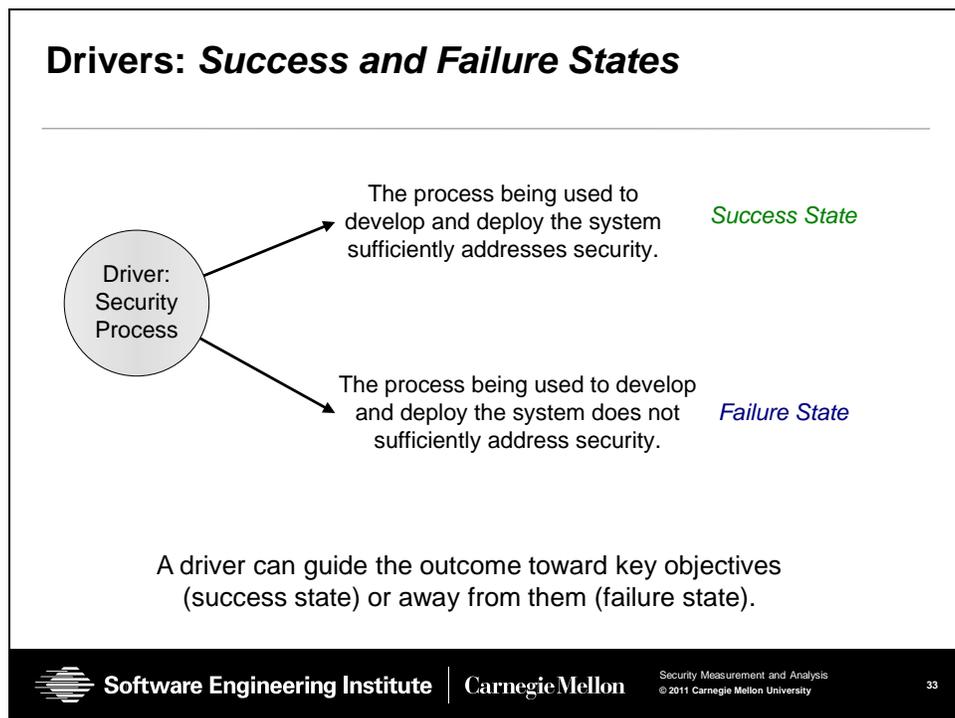
Result

10. Security Requirements
11. Security Architecture and Design
12. Code Security
13. Integrated System Security
14. Adoption Barriers
15. Operational Security Compliance
16. Operational Security Preparedness
17. Product Security Risk Management



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. This slide lists the name of each software security driver. These standard drivers were derived from the software security objective highlighted on slide 21. To date, this set of drivers has not been validated in pilot assessments. The next step in the development of the software security drivers is to validate them through field testing. Once a set of drivers is validated, it serves as an archetype that analysts can quickly tailor and apply to specific programs.

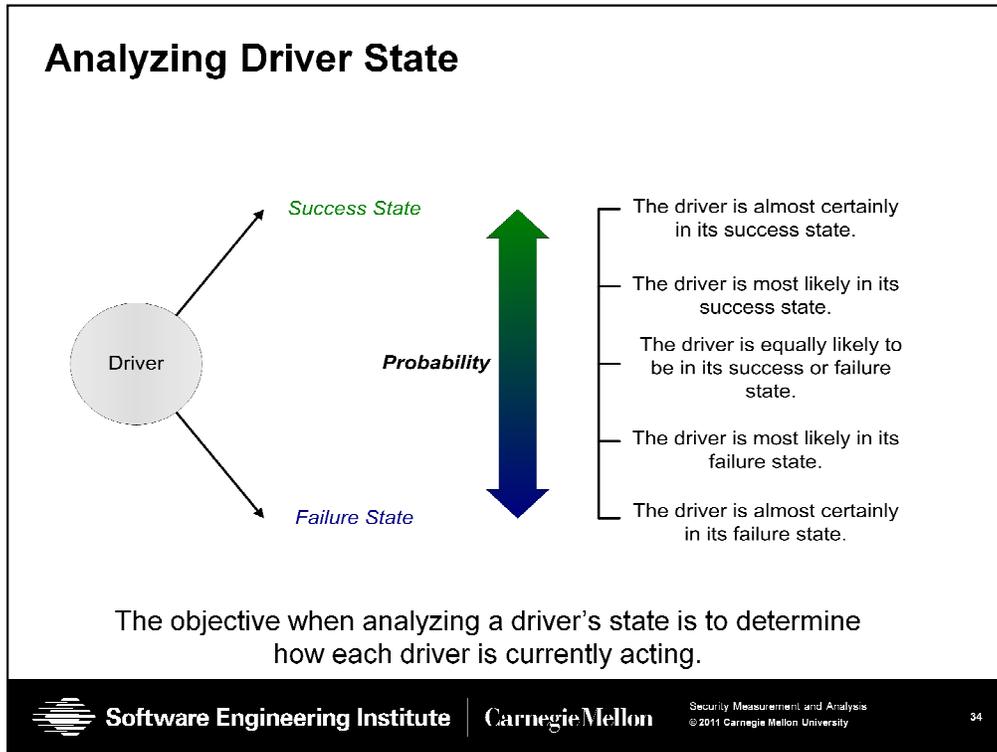




A *driver* is a factor that has a strong influence on the eventual outcome or result (i.e., whether or not objectives will be achieved). Each driver has two possible states: a success state and a failure state. This slide shows the success and failure states for the driver named *Security Process*.

The success state for the Security Process driver 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.

A driver can thus guide the outcome toward key objectives (success state) or away from them (failure state).



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 state
- most likely in its failure state
- almost certainly in its failure state

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

SSR: Evaluating Drivers

Directions: Select the appropriate response to the driver question.

Driver Question	Response
4. Does the process being used to develop and deploy the system sufficiently incorporate security? <i>Consider:</i> <ul style="list-style-type: none"> ▪ Security-related tasks and activities in the program workflow ▪ Conformance to security process models ▪ Measurements and controls for security-related tasks and activities ▪ Process efficiency and effectiveness ▪ Software security development life cycle ▪ Security-related training ▪ Compliance with security policies, laws, and regulations ▪ Security of all product-related information 	<input type="checkbox"/> Yes <input type="checkbox"/> Likely Yes <input type="checkbox"/> Equally Likely <input checked="" type="checkbox"/> Likely No <input type="checkbox"/> No

Driver questions are phrased from the success perspective.

Probability is incorporated into the range of responses for each driver question.

Evidence supporting each response is recorded.



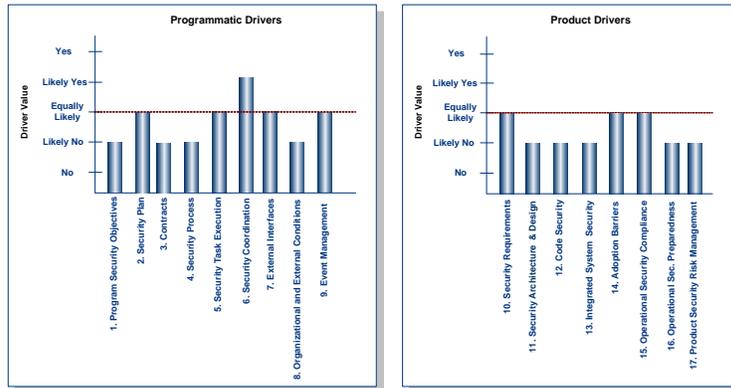
The goal of driver analysis is to determine how each driver is influencing the objectives. More specifically, the probability of success state or failure state for each driver must be established. The driver in this example is evaluated using a yes/no question that is phrased from the *success perspective*. The example on this slide depicts a driver question for the Security Process driver.

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

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



SSR: Driver Profile



A driver profile provides a snapshot of current conditions.
 The driver profile provides a dashboard for program decision makers.



A *driver profile* provides a summary of all drivers relevant to the mission and objectives being assessed. A driver profile can be viewed as a dashboard that provides decision makers with a graphical summary of current conditions and expected performance in relation to the mission and objectives being pursued by a program. This slide provides an example of a driver profile for software security. In the figure, a bar graph is used to show 17 drivers that correspond to the standard set for software security.

The graph depicts the probability that each driver is in its success state. In addition, programmatic drivers are separated from product drivers. The driver profile depicted on this slide indicates that nine drivers are likely in their failure states. These drivers should concern the program’s decision makers.

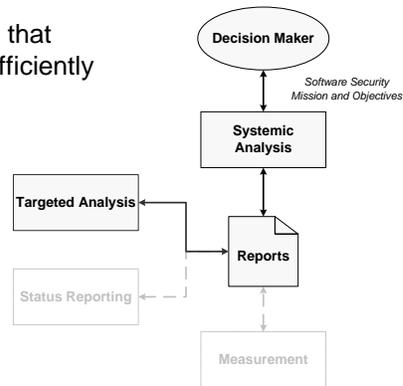


Multi-View Decision Making (MVDM)

Multi-View Decision Making (MVDM) provides a framework for applying multiple management and engineering analysis methods in *distributed management environments*.

MVDM establishes justified confidence that large-scale, networked systems are sufficiently secure to meet operational needs.

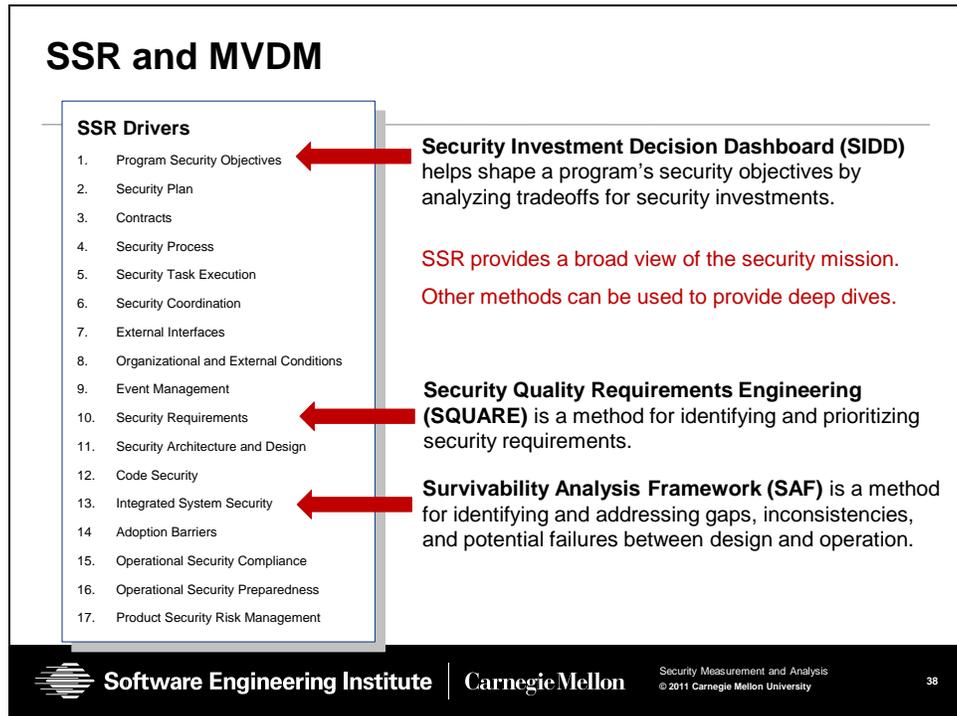
Each MVDM assessment method provides unique insight into the acquisition, development, and operation of large-scale, networked systems.



Multi-View Decision Making (MVDM) provides a framework for applying multiple management and engineering analysis methods in multi-organization environments. A key goal of MVDM is to provide justified confidence that systems of systems are sufficiently secure to meet operational needs.

Each MVDM method provides unique insight into the multi-organization environment that is acquiring, developing, deploying, and operating the system of systems. Together, MVDM's methods present multiple views into the ecosystem, with each view providing decision makers with valuable insights into the ecosystem's current state. MVDM is primarily focused on early life-cycle activities (acquisition, development, and deployment). However, MVDM's current suite of methods also has some applicability during operations and sustainment.

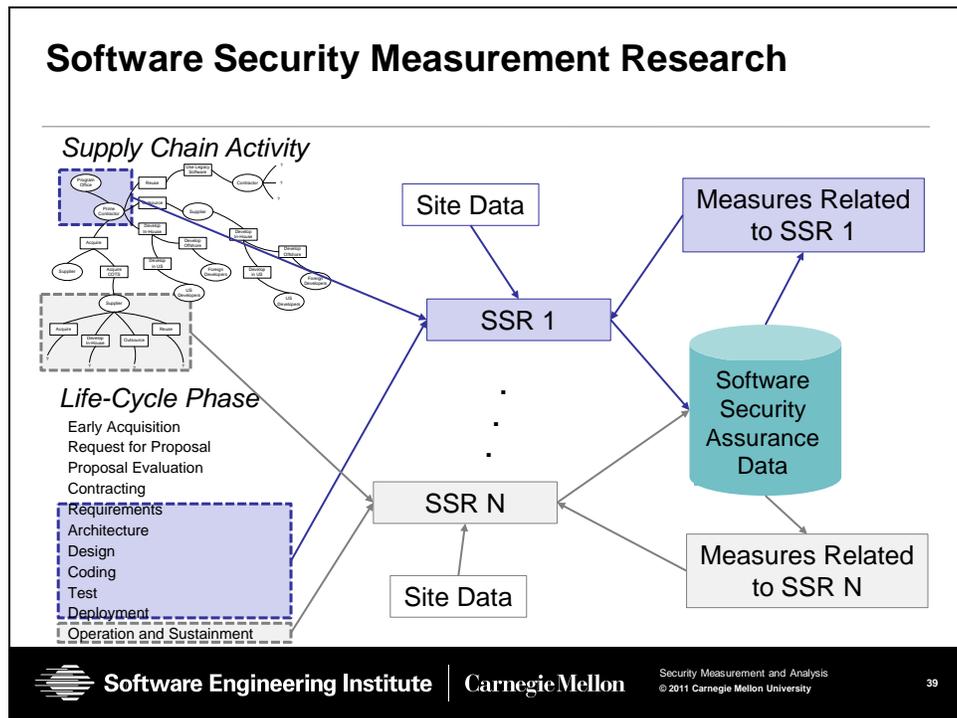




MVDM uses SSR to provide a broad view of software security assurance. An assessment team can use the findings of SSR to select and perform follow-on, “deep-dive” assessments. MVDM helps optimize security assessment activities by applying resources where and when they are most needed.

This slide shows how SSR relates to the following “deep-dive” assessments:

- Security Investment Decision Dashboard (SIDD)
- Security Quality Requirements Engineering (SQUARE)
- Survivability Analysis Framework (SAF)

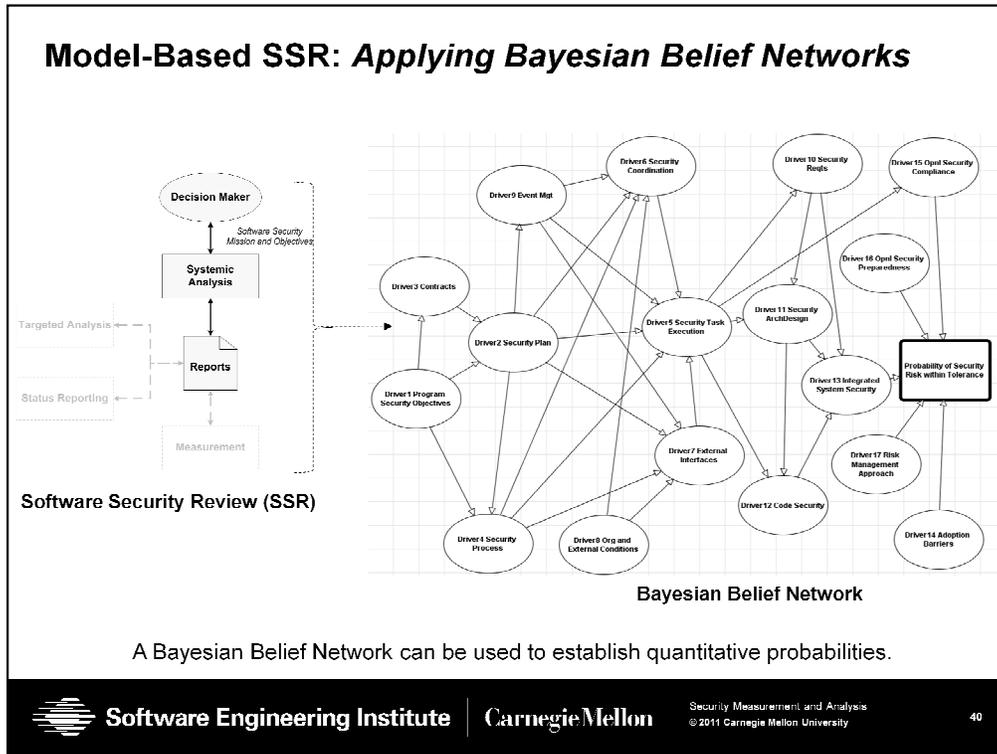


Software Security Measurement Research is initially focused on identifying measures that provide insight into the mission and objectives. As defined here, a measurement is an observation that results in information (reduction of uncertainty) about a quantity. [4]

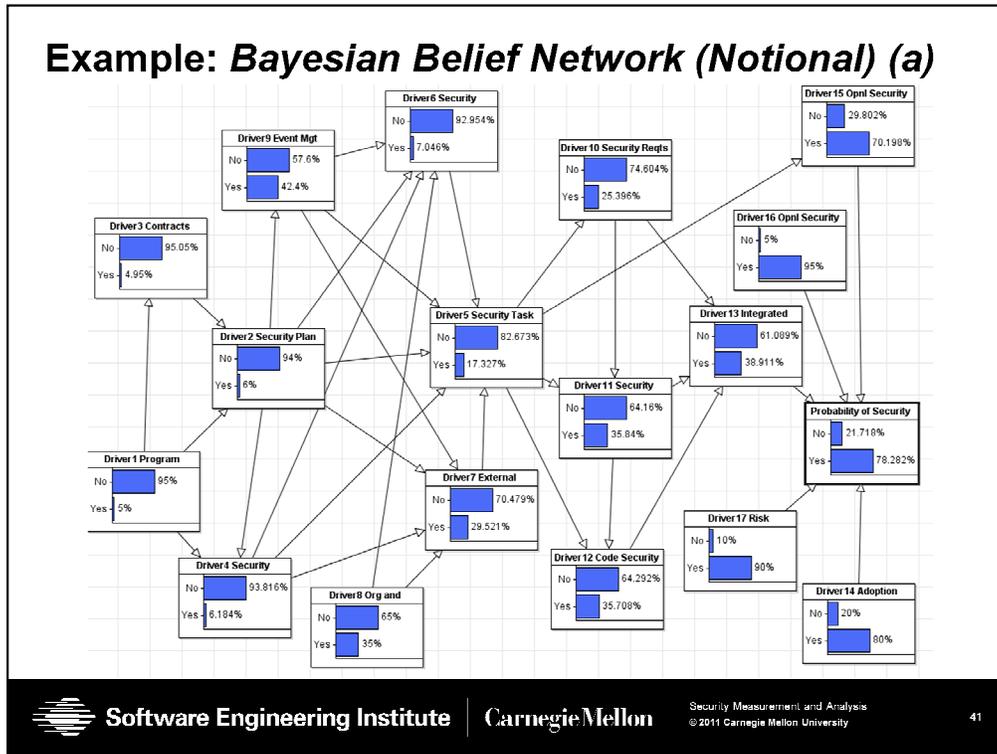
The goal of measurement is to reduce uncertainty in decision making. During SSR, uncertainties related to each driver are identified. Several mechanisms can be employed to reduce uncertainties related to drivers, including measurement, targeted analysis, and status reporting. Here, driver uncertainties can be used to focus a program’s measurement efforts.

The initial focus of this research is analyzing the results of SSR assessments to identify a baseline set of software security measures.

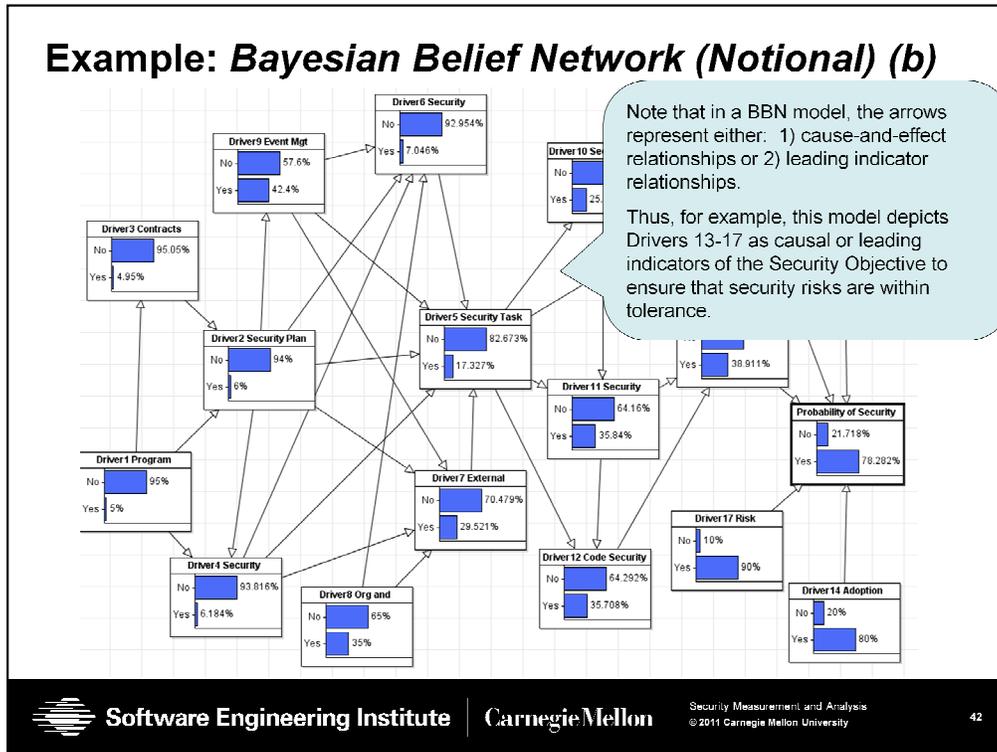
[4] Hubbard, Douglas. *Applied Information Economics Seminar: Executive Overview*. Hubbard Decision Research, 2010. <http://www.hubbardresearch.com/>



Model-Based SSR incorporates predictive analytics, such as Bayesian Belief Networks (BBNs), into its analysis approach. Model-Based SSR enables quantitative analysis of software security assurance using a combination of subjective and objective data.



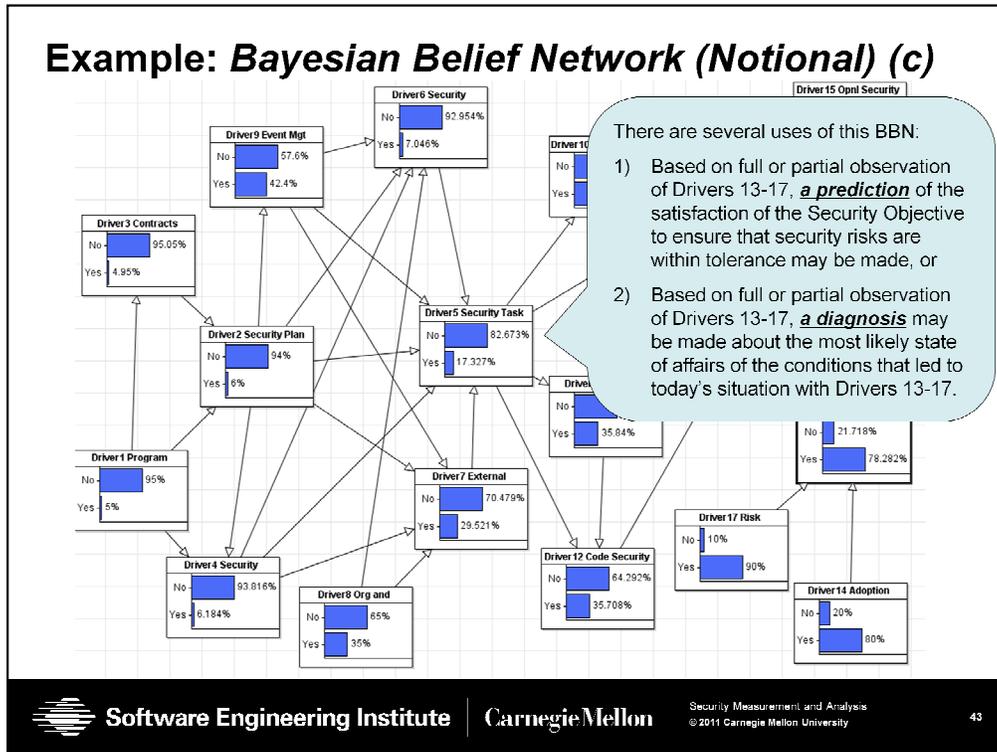
This slide shows a notional BBN based on the 17 drivers for software security.



In a BBN model, the arrows represent either

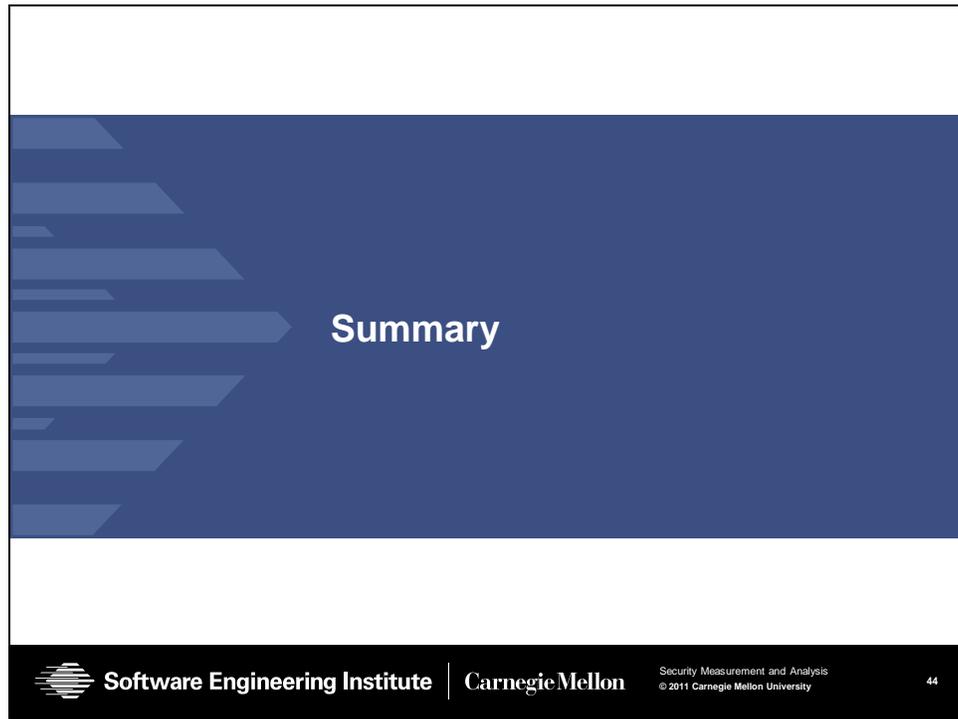
- cause-and-effect relationships
- leading indicator relationships

The example on this slide depicts Drivers 13-17 as causal or leading indicators of the Security Objective (i.e., to ensure that security risks are within tolerance).



This notional BBN can be used in two ways. First, based on full or partial observation of Drivers 13-17, a prediction can be made regarding whether the Security Objective will be achieved.

Second, based on full or partial observation of Drivers 13-17, a diagnosis can be made regarding the most likely state of affairs of the conditions that led to today's situation with Drivers 13-17.



Summary

 **Software Engineering Institute** | **CarnegieMellon** Security Measurement and Analysis
© 2011 Carnegie Mellon University 44

Topic 5: Summary

Summary (a)

Problem:

- Security measurement is insufficient and unproven in large-scale, networked, software-reliant systems across the life cycle and supply chain.

Solution:

- frameworks, methods, and tools that
 - measure and monitor security across the life cycle and supply chain
 - enable systemic analysis of security issues, risks, and uncertainties
 - assess compliance with security standards

Benefits:

- mission-based measurement and analysis of security
- measured confidence in the security of large-scale, networked, software-reliant systems across the life cycle and supply chain
- remediation of security issues, risks, and uncertainties earlier in the life cycle



The problem space for the Security Measurement and Analysis (SMA) Project is:

Security measurement is insufficient and unproven in large-scale, networked, software-reliant systems across the life cycle and supply chain.

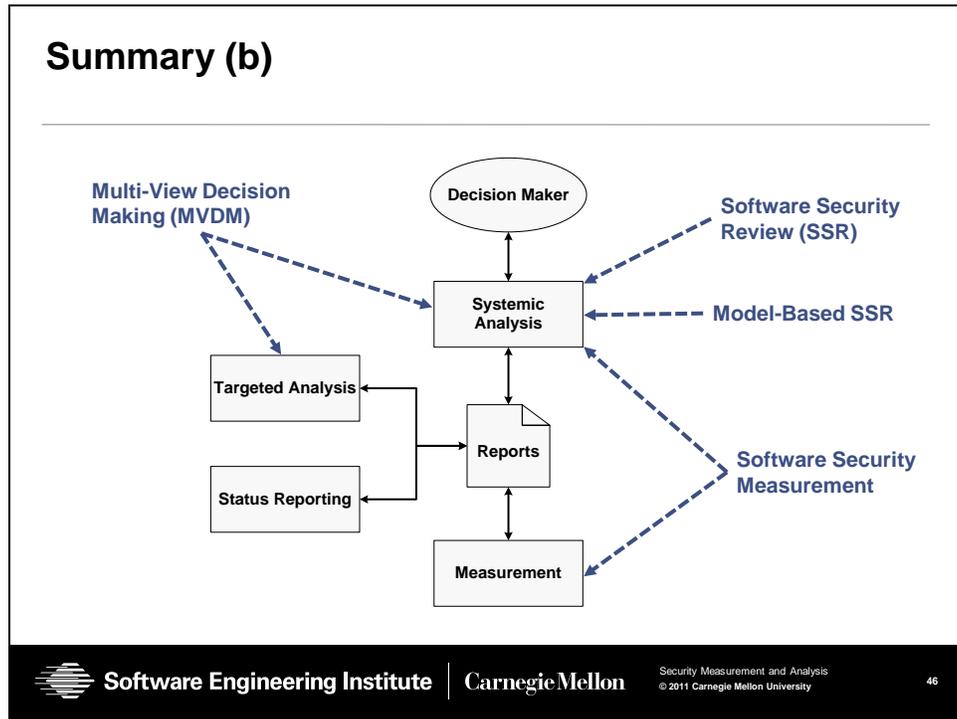
To address this problem space, the SMA Project Team is developing frameworks, methods, and tools that

- measure and monitor security across the life cycle and supply chain
- enable systemic analysis of security issues, risks, and uncertainties
- assess compliance with security standards

Customer benefits from applying SMA frameworks, methods, and tools include the following:

- mission-based measurement and analysis of security
- measured confidence in the security of large-scale, networked, software-reliant systems across the life cycle and supply chain
- remediation of security issues, risks, and uncertainties earlier in the life cycle





This slide summarizes the frameworks and methods currently being developed by the SMA Project Team:

- The SEI Integrated Measurement and Analysis Framework (IMAF) employs systemic 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 large-scale, networked systems.
- The Software Security Review (SSR) is a method conducted by independent teams to assess the security characteristics of software-reliant systems. SSR is a driver-based approach that can be used to measure and monitor software security assurance across the life cycle and supply chain (including acquisition, development, and operations).
- Multi-View Decision Making (MVDM) provides a framework for applying multiple management and engineering analysis methods in multi-organization environments. A key goal of MVDM is to provide justified confidence that systems of systems are sufficiently secure to meet operational needs.
- Model-Based SSR incorporates predictive analytics, such as Bayesian Belief Networks (BBNs), into its analysis approach. Model-Based SSR enables quantitative analysis of software security assurance using a combination of subjective and objective data.
- Software Security Measurement Research is initially focused on identifying measures that provide insight into the mission and objectives. The initial focus of this research is analyzing the results of SSR assessments to identify a baseline set of software security measures.

Research and Development: *Current Needs (a)*

We are looking for customers and collaborators to support work in the following areas:

- **SSR**
 - method refinement
 - development of additional driver sets
 - pilots
- **MVDM**
 - method refinement
 - pilots
- **Model-based SSR**
 - model development
 - pilots



The SEI is looking for customers and collaborators to support work in the following areas:

- SSR – method refinement, development of additional driver sets, and pilots
- MVDM – method refinement and pilots
- Model-based SSR – model development and pilots



Research and Development: *Current Needs* (b)

We are looking for customers and collaborators to support work in the following areas:

- **Software Security Measurement Research**
 - method development for measure identification and validation
 - pilots
- **Training**
 - training development
 - pilots
- **Expansion to Other Security Application Areas**
 - develop methods and tools for selected security application areas (i.e., move beyond *software* security)



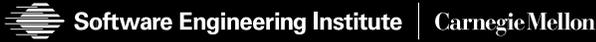
The SEI is looking for customers and collaborators to support work in the following areas:

- Software Security Measurement Research – method development for measure identification and validation and pilots
- Training – training development and pilots
- Expansion to Other Security Application Areas – develop methods and tools for selected security application areas (i.e., move beyond *software* security)





Software Engineering Institute | Carnegie Mellon



Software Engineering Institute | Carnegie Mellon

Security Measurement and Analysis
© 2011 Carnegie Mellon University

49