Software Engineering Institute | Carnegie Mellon University

A Systemic Approach for Assessing Software Supply-Chain Risk

Audrey Dorofee
Carol Woody
Christopher Alberts
Rita Creel
Robert J. Ellison

February 2003

ABSTRACT: In today's business environment, multiple organizations must routinely work together in software supply chains when acquiring, developing, operating, and maintaining software products. The programmatic and product complexity inherent in software supply chains increases the risk that defects, vulnerabilities, and malicious code will be inserted into a delivered software product. As a result, effective risk management is essential for establishing and maintaining software supply-chain assurance over time. The Software Engineering Institute (SEI) is developing a systemic approach for assessing and managing software supply-chain risks. This paper highlights the basic approach being implemented by SEI researchers and provides a summary of the status of this work.

INTRODUCTION

The increasingly global nature of software development has raised concerns that global supply chains could be compromised, allowing malicious code to be inserted into a delivered software product during development or enabling a compromised product to be substituted during delivery or installation. However, while direct attacks on a software supply chain are emerging as a viable possibility, the most likely sources of supply-chain problems continue to be the inadvertent insertion of defects and vulnerabilities into software products. These defects and vulnerabilities affect the ability of software to function as intended during operations and also adversely affect its reliability, security, and safety attributes. Unfortunately, today's complex supply chains tend to increase the probability that software defects and vulnerabilities will be inserted into products, which leads to increased risk during operations.

A key aspect of reducing software supply-chain risk is raising awareness of (1) the conditions that can lead to the insertion of malicious code into software products and (2) the potential for software defects and vulnerabilities to be inadvertently inserted into products. Supply-chain stakeholders need to raise their awareness of these issues by identifying critical software supply-chain risks and developing mitigation plans for those risks.

Software Engineering Institute
Carnegie Mellon University
4500 Fifth Avenue
Pittsburgh, PA 15213-2612

Phone: 412-268-5800
Toll-free: 1-888-201-4479

www.sei.cmu.edu

The Carnegie Mellon® Software Engineering Institute (SEI) has chartered the Software Supply-Chain Project within its CERT® Program. The project's goal is to develop an approach for assessing software supply chains and raising awareness of the associated risks. The goal of project is to enable a software supply-chain's stakeholders to maintain risk within a reasonable tolerance over time and, as a result, establish sufficient assurance that the software supply chain is in position to achieve its objectives. This paper highlights the basic approach being implemented by SEI researchers to assess software-related risk throughout the supply chain and provides a summary of the status of this work.

SOFTWARE SUPPLY CHAIN

A supply chain is defined as the set of suppliers that contribute to the content of a product or system (both hardware and software) or that have the opportunity to modify its content [Ellison 2010]. The SEI Software Supply-Chain Project is primarily focused on the software supply chain, which is defined as the network of stakeholders that contribute to the content of a software product or that have the opportunity to modify its content. Figure 1 illustrates the notion of the software supply chain.

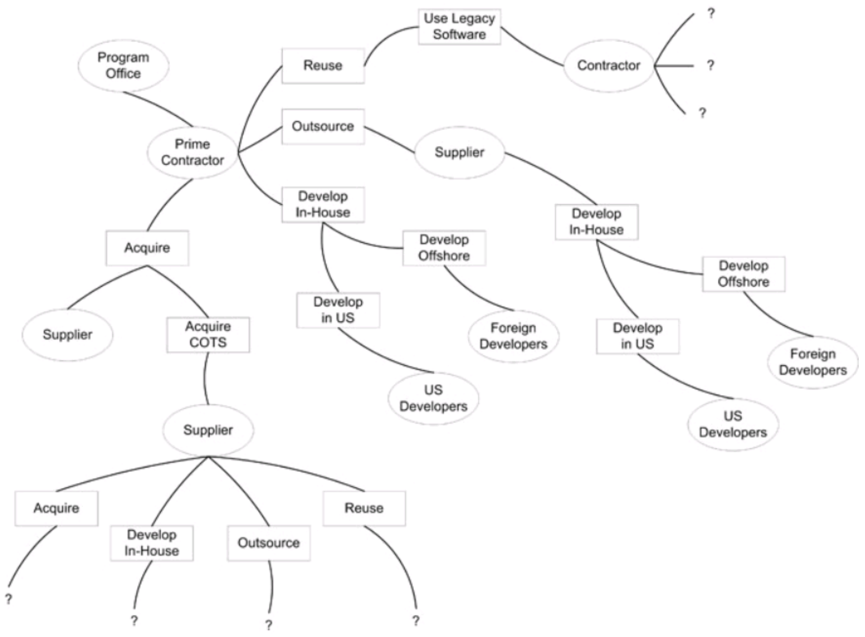


Figure 1: Software Supply Chain

The purpose, or mission, of a software supply chain is to deploy software products that function as intended and are reliable, safe, and secure. Figure 1 shows that a software supply chain comprises multiple organizations. In some cases, relationships among organizations are formally defined. For example, an acquirer can execute a formal contract with a supplier that governs the relationship between the two organizations. Typically, the acquirer provides a set of requirements, and the supplier develops a software product that meets those requirements. Another example of a formal agreement between organizations is when an acquirer licenses commercial off-the-shelf (COTS) software from a supplier. A license outlines any terms and conditions regarding the acquirer's use of the software product. However, the supplier is free to update or make changes to the software as it sees fit, without considering the impact on its customers (i.e., its licensees).

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

In general, no single administrative structure or set of policies governs all organizations participating in a software supply chain. In addition, no single manager has authority over all organizations within the supply chain. Multiple points of management control exist, which creates a degree of programmatic complexity that can be difficult to manage effectively. In addition to managing programmatic complexity, software supply-chain stakeholders also need to address the growing complexity of software products.

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

- managerial independence – The management of each system within a system of systems is independent from the management of the other systems.
- operational independence – Each system within a system of systems provides useful functionality apart from other systems.
- evolutionary character – Each system within a system of systems grows and changes independently of other systems over time.

- emergent behavior – Certain behaviors of a system of systems arise from the interactions among the individual systems and are not embodied in any of the individual systems.
- geographic distribution – Individual systems within a system of systems are dispersed over large geographic areas.

A software supply chain is an example of a system-of-systems environment, where multiple, independently managed organizations provide technical capabilities via a set of interdependent, networked systems. Software supply-chain assurance is defined as justified confidence that a software product functions as intended and is reliable, safe, and secure. The programmatic and product complexity inherent in software supply chains tends to increase its risk, and a high degree of risk corresponds to low assurance. Effective risk management is thus essential for establishing and maintaining software supply-chain assurance over time.

SOFTWARE SUPPLY-CHAIN RISK

The term risk is used universally, but different audiences often attach different meanings to it [Kloman 1990]. In fact, the details about risk and how it supports decision making depend upon the context in which it is applied [Charette 1990]. For example, safety professionals view risk management in terms of reducing the number of accidents and injuries. A hospital administrator views risk as part of the organization's quality assurance program, while the insurance industry relies on risk management techniques when setting their rates. Each industry thus uses a definition that is uniquely tailored to its perspective and context. As a result, no universally accepted definition of risk exists.

However, whereas specific definitions of risk might vary, a few characteristics are common to all definitions. In fact, for risk to exist in any circumstance, the following three conditions must be satisfied [Charette 1990]:

1. The potential for loss must exist.
2. Whether the risk will occur is not known with certainty; however a probability of occurrence can be determined.
3. Some choice or decision is required to deal with the risk.

These characteristics can be used to forge a very basic definition of the word risk. Most definitions focus on the first two conditions—loss and probability—because they are the two quantifiable aspects of risk. Bearing this in mind, the essence of risk, no matter what the domain, can be succinctly captured by the following definition: Risk is the likelihood of suffering loss [Dorofee 1996]. Put another way, risk is a measure of the likelihood that a threat will lead to a loss coupled with the magnitude of the loss.

Programmatic and product interdependencies are forcing software supply-chain stakeholders to rethink how they manage risk. Each group within the supply chain can no longer assume a parochial view of risk, where its risks are considered to be isolated from those of other groups. They must view risks across the software supply chain as being interdependent. As a result of these interdependencies among risks, stakeholders are being forced to assess and manage risk within a broader system-of-systems context.

Of particular concern is the risk that products produced by a software supply chain will fail to function as intended or fail to operate in a reliable, safe, and secure manner. Many sources can trigger risks to a software supply-chain's products, including

- acquirer actions – requirements, proposal, and source selection issues triggered by an acquirer
- supplier actions – architecture, design, and coding defects initiated by a supplier
- supply-chain logistics – inadequate access control of products or services in each step of the supply chain (e.g., failures in supply-chain component delivery or configuration control)
- product – defects and issues associated with a software product
- operation and sustainment – operational issues arising from changes in how a fielded product or service is used and maintained over time

For example, consider how product risk can affect a supply chain. An acquirer typically has limited information about a software product's security characteristics when purchasing COTS software. If the COTS software contains significant security vulnerabilities, it is considered to be a high-risk component. However, COTS software does not operate in isolation when it is part of a software supply chain. In the supply-chain context, high-risk COTS software is networked with other software products and systems to produce an integrated software-reliant system or system of systems. As a result, the software supply chain inherits risk from its constituent software products and systems.

Consider a second example that is focused on operation-and-sustainment risk. Software products typically provide more functionality than users need. In many instances, unused features and services are enabled during operations. When unused functionality is enabled, it can lead to security vulnerabilities and increase operational security risk. In general, all unnecessary functionality should be disabled to minimize the potential for cyber attacks during operations.

These examples begin to show the breadth of software supply-chain activities. In fact, software supply chains span all phases of the acquisition life cycle, beginning with early acquisition activities and continuing through system retirement. To ensure that software supply-chain risks are managed appropriately, stakeholders must assume a comprehensive life-cycle approach when developing a software supply-chain risk assessment.

APPROACHES FOR ASSESSING RISK

Risk management defines an approach for minimizing exposure to potential losses by providing a disciplined environment for [Alberts 2010]

- continuously assessing what could go wrong (i.e., assessing risks)
- determining which risks to address (i.e., setting mitigation priorities)
- implementing actions to address high-priority risks and bring those risks within tolerance

Assessment is a fundamental aspect of risk management. It provides a foundation for ensuring that risk is maintained within an acceptable tolerance over time by identifying circumstances that might lead to harm or loss. When a software supply chain's risk is within an acceptable tolerance, stakeholders have reasonable assurance that the supply chain is in position to achieve its objectives. Risk assessment is thus an important part of software supply-chain assurance.

A software supply chain is an example of an interactively complex socio-technical system that spans multiple organizational entities. Here, a socio-technical system is defined as interrelated technical and social elements that are engaged in goal-oriented behavior. Elements of a socio-technical system include the people who are organized in teams or departments to do their work tasks and the technical systems on which people rely when performing work tasks.

Two classes of risk assessments can be used when evaluating interactively complex socio-technical systems: (1) system decomposition and event analysis and (2) systemic analysis [Alberts 2009]. Both classes will be examined in this section, beginning with system decomposition and event

SYSTEM DECOMPOSITION AND EVENT ANALYSIS

Most traditional risk assessments are based on the principle of system decomposition and event analysis [Alberts 2009]. The first step when conducting this type of analysis is to decompose the socio-technical system into its constituent parts, or components. Individual components are then prioritized, and a subset of components is designated as being critical. Next, analysts evaluate how a series of predefined events might affect each critical component, estimating the likelihood that each event will occur and the magnitude of the resulting loss. Risk is ultimately assessed using a measure called risk exposure, which is defined as the product of an event's likelihood of occurrence and the magnitude of loss if the event were to occur. The main goal of system decomposition and event analysis is to ensure that each risk is acceptable to stakeholders.

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

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

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

SYSTEMIC ANALYSIS

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

- influences of environmental factors

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

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

SEI experience shows that systemic analysis is useful for understanding the behavior of interactively complex socio-technical systems because it effectively handles the high degree of uncertainty that is inherent in these systems [Alberts 2009]. Applying systemic analysis to interactively complex socio-technical systems provides the decision maker with a means of confidently assessing the behavior of the system as a whole, which is necessary when assessing assurance. As a result, the SEI is implementing a systemic approach for assessing software supply-chain risks across the life cycle.

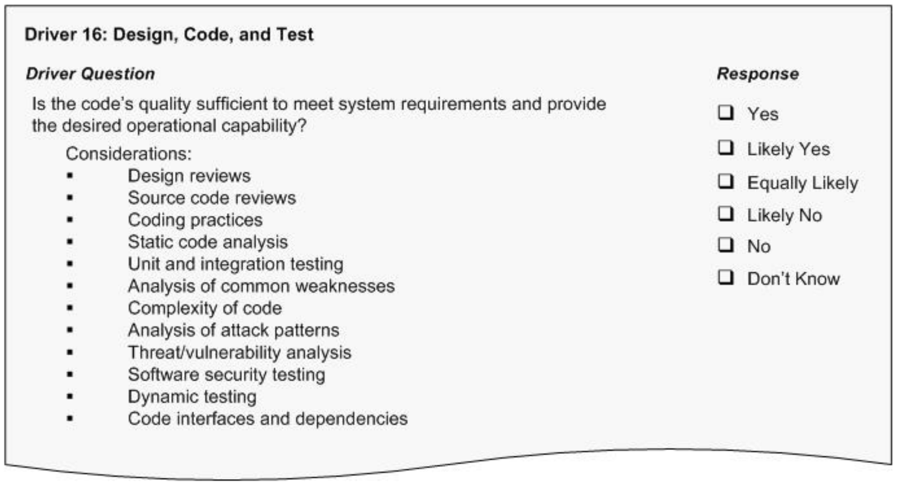
SOFTWARE SUPPLY-CHAIN RISK ASSESSMENT

As mentioned in the previous section, a systemic risk assessment is based on a small set of factors, called drivers, that strongly influence the eventual outcome or result. SEI experience shows that approximately 15-25 drivers are needed to establish a comprehensive profile of systemic risks to mission success [Alberts 2009]. Each driver is represented as a yes-no question, where an answer of yes means that the driver is in its success state (i.e., contributing minimal risk to the software supply-chain mission) and an answer of no means that the driver is in its failure state (i.e., contributing a severe degree of risk to the software supply-chain mission). The following is a listing of the standard set of software supply-chain drivers:

1. Software Supply-Chain Objectives
2. Plan
3. Contracts
4. Process
5. Task Execution

6. Coordination
7. Software Supply-Chain Interfaces
8. Information Management
9. Technology
10. Facilities and Equipment
11. Environmental Conditions
12. Compliance
13. Event Management
14. Requirements
15. Architecture
16. Design, Code, and Test
17. System Functionality
18. System Integration
19. Operational Support
20. Adoption Barriers
21. Operational Preparedness
22. System Risk Tolerance
23. Certification and Accreditation
24. Sustainment

The 24 drivers in the set provide a comprehensive profile of software supply-chain risks. The drivers are aligned with the sources of software supply-chain risks outlined in Section 3 of this paper (acquirer actions, supplier actions, supply-chain logistics, product, and operation and sustainment). An example of a driver question is shown in Figure 2.



Driver 16: Design, Code, and Test

Driver Question

Is the code's quality sufficient to meet system requirements and provide the desired operational capability?

Considerations:

- Design reviews
- Source code reviews
- Coding practices
- Static code analysis
- Unit and integration testing
- Analysis of common weaknesses
- Complexity of code
- Analysis of attack patterns
- Threat/vulnerability analysis
- Software security testing
- Dynamic testing
- Code interfaces and dependencies

Response

☐ Yes
☐ Likely Yes
☐ Equally Likely
☐ Likely No
☐ No
☐ Don't Know

Figure 2: Question and Considerations for the "Design, Code, and Test" Driver

The driver question depicted in the figure is named Design, Code, and Test (number 16 in the standard set of 24 drivers). This particular driver is focused on the quality of the code developed by a software supply chain. The considerations listed below the question provide guidance about which sources of information are relevant to the driver question.

Software Supply-Chain Risk Assessment Process

The SEI is developing a risk assessment designed to evaluate software supply-chain drivers and establish a risk profile for a software supply chain. Figure 3 illustrates the SEI's prototype, driver-based approach for assessing systemic software supply-chain risks.

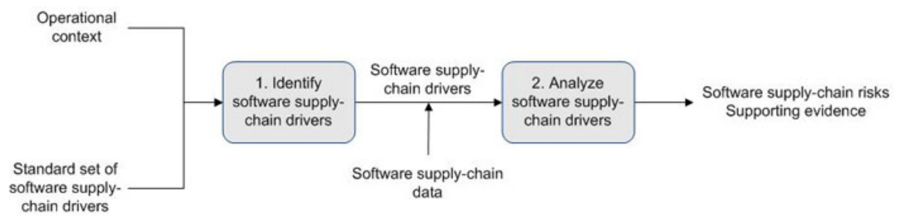

Operational context
Standard set of software supply-chain drivers
1. Identify software supply-chain drivers
Software supply-chain drivers
Software supply-chain data
2. Analyze software supply-chain drivers
Software supply-chain risks
Supporting evidence

Figure 3: Software Supply-Chain Risk Assessment

As shown in the figure, the software supply-chain assessment comprises two main activities: (1) Identify software supply-chain drivers and (2) Analyze software supply-chain drivers. Each activity will be described briefly, beginning with identifying software supply-chain drivers.

Identify Software Supply-Chain Drivers

The goal of the assessment's first activity, Identify software supply-chain drivers, is to establish a set of drivers for a software supply chain. Identifying a set of drivers draws on the following two inputs: (1) a standard set of software supply-chain drivers and (2) operational context. The standard set of software supply-chain drivers comprises a collection of critical factors required for supply-chain mission success. (Refer to Section 5 for a list of the standard set of software supply-chain drivers.)

To ensure that it is useful for a given software supply chain, the standard set of drivers must be tailored to the needs and requirements of that supply chain. The second input, operational context, includes data about the mission, objectives, requirements, and environment related to the specific software supply chain that is being assessed. Operational context is used to tailor the standard set of drivers to meet the needs of the specific software supply chain that is being assessed. This tailoring produces a set of drivers that are uniquely applicable to a given software supply chain. The tailored set of drivers is analyzed during the second activity, which is described next.

Analyze Software Supply-Chain Drivers

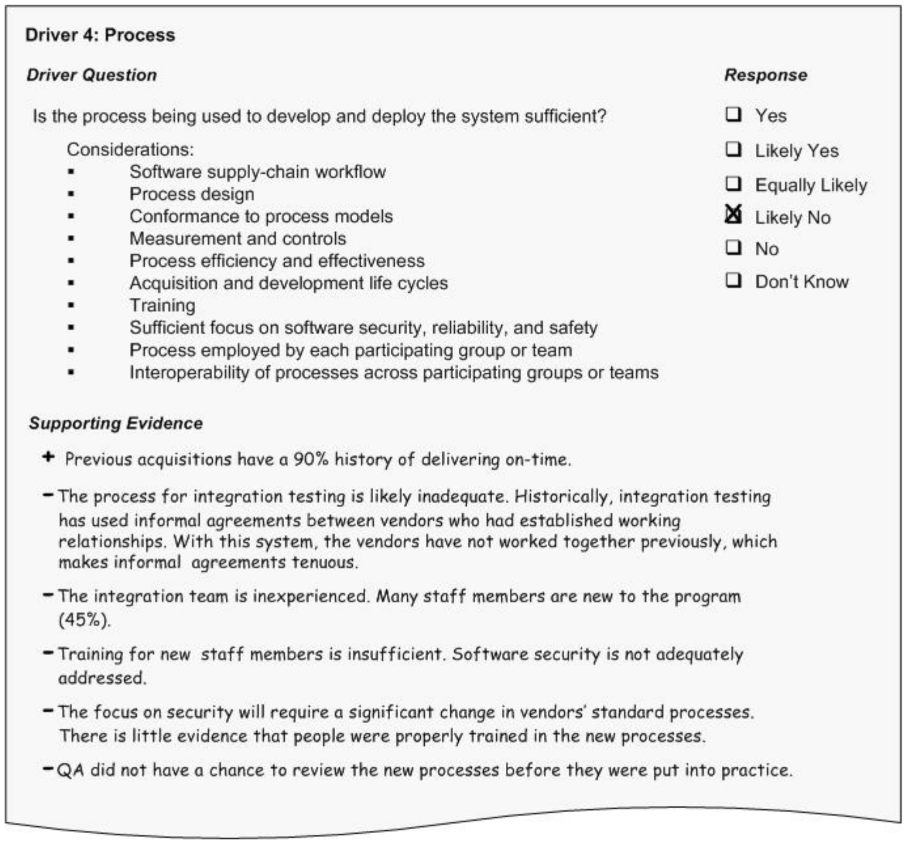
The goal of the second assessment activity, Analyze software supply-chain drivers, is to evaluate how drivers are influencing the supply chain's core mission. Two inputs are required when analyzing software supply-chain drivers: (1) software supply-chain drivers and (2) software supply-chain data.

The first input, software supply-chain drivers, is generated by the first assessment activity. The second input, software supply-chain data, is obtained by conducting interviews with people who are knowledgeable about the supply chain being assessed and by reviewing documentation that is relevant to that supply chain. The risk produced by each driver is then analyzed, and evidence supporting that analysis is documented. For example, analyzing the Design, Code, and Test driver shown in Figure 2 would require examining results of design reviews, conducting source code reviews, and analyzing common weaknesses, among other sources of evidence.

Example

Consider the driver question depicted in Figure 4; it illustrates the question, considerations, and supporting evidence for the Process driver (number 4 in the standard set of 24 drivers). Answering a driver question requires examining how conditions and potential events are affecting that driver. The goal is to determine if the driver is

- almost certainly in its success state (response of yes)
- most likely in its success state (response of likely yes)
- equally likely in its success or failure states (response of equally likely)
- most likely in its failure state (response of likely no)
- almost certainly in its failure state (response of no)



Driver 4: Process

Driver Question — ***Response***

Is the process being used to develop and deploy the system sufficient?

Considerations:

- Software supply-chain workflow
- Process design
- Conformance to process models
- Measurement and controls
- Process efficiency and effectiveness
- Acquisition and development life cycles
- Training
- Sufficient focus on software security, reliability, and safety
- Process employed by each participating group or team
- Interoperability of processes across participating groups or teams

Response: ☐ Yes ☐ Likely Yes ☐ Equally Likely [✓ Likely No] ☐ No ☐ Don't Know

Supporting Evidence

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

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

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

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

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

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

Figure 4: An Analyzed Driver

When answering driver questions, analysts must document evidence to support their answers. The data generated by interviews and document reviews form the basis for the supporting evidence that is used to justify analysts' responses to driver questions. Refer to Figure 4 for an example of supporting evidence that has been documented for the Process driver. The response of likely no for the Process driver indicates a high degree of risk to the software supply-chain mission.

Development Approach

The SEI has conducted research and development in the area of risk management since the early 1990s. The SEI has applied risk management methods, tools, and techniques across the life cycle (including acquisition, development, and operations). In addition, past SEI research examined various types of risk, including software development risk [Dorofee 1996, Williams 1999, Alberts 2009], system acquisition risk [Gallagher 1999], operational risk [Gallagher 2005], information security risk [Alberts 2002], and systemic risk [Alberts 2009], among others. The Software Supply-Chain Project is leveraging previous SEI work in risk management, particularly its work in assessing systemic risk in system-of-systems environments [Alberts 2005].

1. The SEI approach for developing a software supply-chain risk assessment includes the following activities:
2. Tailor the SEI method for assessing systemic risk in a system-of-systems environment for application to software supply chains.
3. Develop a prototype set of drivers for software supply chains based on the experience and expertise of the SEI project team.
4. Review the prototype set of drivers with external experts and refine the set as appropriate.
5. Pilot the assessment method with selected software supply chains and refine the method based on the results of the pilots.

To date, the SEI project team has completed the first two steps in the development approach. The team is beginning to engage external experts to get their feedback and comments regarding the drivers. The prototype set of drivers will be updated and refined based on the feedback obtained from the external experts.

Summary

A software supply chain is the network of stakeholders that contribute to the content of a software product or that have the opportunity to modify its content. The SEI Software Supply-Chain Project is developing an approach for assessing software supply-chain risks. The overarching goal of the project is to assure that a software supply chain is in position to achieve its objectives by effectively managing its risk.

While many people consider risk management to be a relatively mature discipline, programmatic and product complexity is forcing software supply-chain stakeholders to rethink how they manage their risk. Each group within a software supply chain can no longer assume that its risks are isolated from those of their collaborators and partners. Each participant in a supply chain must view risks across the software supply chain as being interdependent. As a result, software supply-chain risks must be assessed and managed within a system-of-systems context.

SEI experience shows that systemic analysis of risk is useful when analyzing performance in system-of-systems environments, such as supply chains. Applying systemic risk analysis to a software supply chain provides decision makers with a means of confidently assessing the behavior of the supply chain as a whole, which is necessary when assessing assurance.

REFERENCES

[Alberts 2002]

Alberts, Christopher & Dorofee, Audrey. Managing Information Security Risks: The OCTAVESM Approach. Boston, MA: Addison-Wesley, 2002 (ISBN 0-321-11886-3).

[Alberts 2005]

Alberts, Christopher & Dorofee, Audrey. Mission Assurance Analysis Protocol (MAAP): Assessing Risk in Complex Environments (CMU/SEI-2005-TN-032). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005.

[Alberts 2009]

Alberts, Christopher & Dorofee, Audrey. A Framework for Categorizing Key Drivers of Risk (CMU/SEI-2009-TR-007). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2009.

[Alberts 2010]

Alberts, Christopher & Dorofee, Audrey. Risk Management Framework (CMU/SEI-2010-TR-017). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2010.

[Charette 1990]

Charette, Robert N. Application Strategies for Risk Analysis. New York, NY: McGraw-Hill Book Company, 1990.

[Dorofee 1996]

Dorofee, Audrey, Walker, Julie, Alberts, Christopher, Higuera, Ron, Murphy, Richard, & Williams, Ray.Continuous Risk Management Guidebook. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1996.

[Ellison 2010]

Ellison, Robert & Woody, Carol. “Supply-Chain Risk Management: Incorporating Security into Software Development.” Proceedings of the 43rd Hawaii International Conference on System Sciences (HICSS) (CD-ROM), January 5-8, 2010, Computer Society Press, 2010 (10 pages).

[Gallagher 1999]

Gallagher, Brian. Software Acquisition Risk Management Key Process Area (KPA): A Guidebook Version 1.02 (CMU/SEI-99-HB-001). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999.

[Gallagher 2005]

Gallagher, Brian, Case, Pamela, Creel, Rita, Kushner, Susan, & Williams, Ray. A Taxonomy of Operational Risks (CMU/SEI-2005-TN-036). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005.

[Kloman 1990]

Kloman, Henry Felix. "Risk Management Agonists." Risk Analysis 10, 2 (June 1990): 201-205.

[Leveson 2004]

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

[Levine 2003]

Levine, Linda, Meyers, B. Craig, Morris, Ed, Place, Patrick R. H., & Plakosh, Daniel. Proceedings of the System of Systems Interoperability Workshop (February 2003) (CMU/SEI-2003-TN-016). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003.

[Maier 1996]

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

[Williams 1999]

Williams, Ray, Pandelios, George, & Behrens, Sandra. Software Risk Evaluation (SRE) Method Description (Version 2.0) (CMU/SEI-99-TR-029). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999.