Chapter 7. Governance, and Managing for More Secure Software

7.1. Introduction

The objective of this chapter is to help software project managers (1) more effectively engage their leaders and executives in security governance and management by understanding how to place security in a business context and (2) better understand how to enhance their current management practices and thereby produce more secure software. Armed with this material, managers can become attentive, security-conscious leaders who are in a better position to make well-informed security investment decisions. With this support, managers can then take actionable steps to implement effective security governance and management practices across the software and system development life cycle.

Governance and management of security are most effective when they are systemic—that is, woven into the culture and fabric of organizational behaviors and actions. In this regard, culture is defined as the predominating shared attitudes, values, goals, behaviors, and practices that characterize the functioning of a group or organization. Culture creates and sustains connections among principles, policies, procedures, processes, products, people, and performance. Effective security should be thought of as an attribute or characteristic of an organization. It becomes evident when all members of the organization proactively carry out their roles and responsibilities, creating a culture of security that displaces ignorance and apathy.

To achieve this outcome, security must come off the technical sidelines, abandoning its traditional identity of "activities and responsibilities solely relegated to software development and IT departments." Today,

boards of directors, senior executives, and managers all must work to establish and reinforce a relentless, ongoing drive toward effective enterprise, information, system, and software security. If the responsibility for these tasks is assigned to roles that lack the authority, accountability, and resources to implement and enforce them, the desired level of security will not be articulated, achieved, or sustained.

Contrary to the popular belief that security is a technical issue, even the best efforts to buy secure software and build security into developed software and operational systems encounter "considerable resistance because the problem is mostly organizational and cultural, not technical" **[Steven 2006]**. Software and information security are about spending money where the definition of success is "nothing bad happens." As time goes on, this kind of effort can become a tough sell to business leaders as the "We haven't been attacked lately, so we can cut back on spending" mentality sets in.

Project managers need to elevate software security from a stand-alone, technical concern to an enterprise issue. Because security is a business problem,[1] the organization must activate, coordinate, deploy, and direct many of its core resources and competencies to manage security risks in concert with the entity's strategic goals, operational criteria, compliance requirements, and technical system architecture. To *sustain* enterprise security, the organization must move toward a security management process that is strategic, systematic, and repeatable, with efficient use of resources and effective, consistent achievement of goals **[Caralli 2004b]**.

E M L L3 L4

7.2. Governance and Security[*]

Governance entails setting clear expectations for business conduct and then following through to ensure the organization fulfills those expectations. Governance action flows from the top of the organization to all of its business units and projects. Done right, governance facilitates an organization's approach to nearly any business problem, including security. National and international regulations call for organizations—and

their leaders—to demonstrate due care with respect to security. This is where governance can help.

7.2.1. Definitions of Security Governance

The term *governance* applied to any subject can have a wide range of interpretations and definitions. For the purpose of this chapter, we define governing for enterprise security[2] as follows:

Directing and controlling an organization to establish and sustain a culture of security in the organization's conduct (beliefs, behaviors, capabilities, and actions)

Treating adequate security as a non-negotiable requirement of being in business **[Allen 2005]**

In its publication *Information Security Handbook: A Guide for Managers* **[Bowen 2006]**, NIST defines information security governance as:

> The process of establishing and maintaining a framework and supporting management structure and processes to provide assurance that information security strategies

– are aligned with and support business objectives,

– are consistent with applicable laws and regulations through adherence to policies and internal controls, and

– provide assignment of responsibility,

> all in an effort to manage risk.

In his article "Adopting an Enterprise Software Security Framework," John Steven, a Principal at Cigital, states

> In the context of an Enterprise Software Security Framework, governance is competency in measuring software-induced risk and supporting an objective decision-making process for remediation and software release. This competency involves creating a seat at

the project management table for software risk alongside budget and scheduling concerns **[Steven 2006]**.

(See also **Section 7.5**.)

In the context of security, governance incorporates a strong focus on risk management. Governance is an expression of responsible risk management, and effective risk management requires efficient governance. One way governance manages risk is to specify a framework for decision making. It makes clear who is authorized to make decisions, what the decision making rights are, and who is accountable for decisions. Consistency in decision making across an enterprise boosts confidence and reduces risk.

7.2.2. Characteristics of Effective Security Governance and Management

One of the best measures that an organization is addressing security as a governance and management concern is a consistent and reinforcing set of beliefs, behaviors, capabilities, and actions that match up with security best practices and standards. These measures aid in building a security-conscious culture. They can be expressed as statements about the organization's current behavior and condition:[3]

• Security is managed as an enterprise issue, horizontally, vertically, and cross-functionally throughout the organization. Executive leaders understand their accountability and responsibility with respect to security for the organization; for their stakeholders; for the communities they serve, including the Internet community; and for the protection of critical national infrastructures and economic and national security interests.

• Security is treated as a business requirement. It is considered a cost of doing business and perceived as an investment rather than an expense or a discretionary budget-line item. Security policy is set at the top of the organization with input from key stakeholders. Business units and staff are not allowed to decide unilaterally how much security they want. Adequate and sustained funding and allocation of adequate security resources are a given.

• Security is considered an integral part of normal strategic, capital, project, and operational planning cycles. Security has achievable, measurable objectives that are integrated into strategic and project plans and implemented with effective controls and metrics. Reviews and audits of plans identify security weaknesses and deficiencies and requirements for the continuity of operations and measure progress against plans of action and milestones.

• Security is addressed as part of any new project initiation, acquisition, or relationship and as part of ongoing project management. Security requirements are addressed throughout all system/software development life-cycle phases, including acquisition, initiation, requirements engineering, system architecture and design, development, testing, operations, maintenance, and retirement.

• Managers across the organization understand how security serves as a business enabler. They view security as one of their responsibilities and understand that their team's performance with respect to security is measured as part of their overall performance.

• All personnel who have access to digital assets and enterprise networks understand their individual responsibilities to protect and preserve the organization's security, including the systems and software that it uses and develops. Awareness, motivation, and compliance are the accepted, expected cultural norm. Rewards, recognition, and consequences with respect to security policy compliance are consistently applied and reinforced.

Leaders who are committed to dealing with security at a governance level can use this checklist to determine the extent to which a security-conscious culture is present (or needs to be present) in their organizations. The relative importance of each statement depends on the organization's culture and business context.

In the next section, which was originally published as an article in *IEEE Security & Privacy*, John Steven explains which governance and management actions to take to address software security at the enterprise level.

Example: Bank of America

Rhonda MacLean, (former) chief information security officer at Bank of America, describes the bank's approach to enterprise security at both a governance and management level:

> On a structural level, Bank of America has established a security compliance framework that includes commitment and accountability, policies and procedures, controls and supervision, regulatory oversight, monitoring, training and awareness, and reporting. Bank of America has also established a four-level information security governance model that maps out the responsibilities of board directors, business executives, chief information officers, corporate audit, the security department, legal, corporate and line-of-business, privacy, and supply chain management.
> The board of directors is responsible for reviewing the corporate information security program and policy, while senior management is accountable for ensuring compliance with applicable laws, regulations, and guidelines and for establishing compliance roles, accountabilities, performance expectations, and metrics. It's up to the auditors to ensure the commitment and accountability for information security controls.
> Bank of America's corporate information security department focuses on people, technology, and processes using a protect/detect/respond–recover model and measures its progress based on the Six Sigma quality methodology. Bank of America measures security based on failed customer interactions rather than on downtime, performance, or the number of infections or attacks. Achieving 99 percent uptime isn't important if the 1 percent downtime impacts 30 million customers **[McCollum 2004]**.

E M L L3

7.3. Adopting an Enterprise Software Security Framework[4]

Most organizations no longer take for granted that their deployed applications are secure. But even after conducting penetration tests, network and hosting security personnel spend considerable time chasing incidents. Your organization might be one of the many that have realized the "secure the perimeter" approach doesn't stem the tide of incidents because the software it's building and buying doesn't resist attack.

Painfully aware of the problem, savvy organizations have grappled with how to build security into their software applications for a few years now. Even the best efforts have met considerable resistance because the problem is mostly organizational and cultural, not technical—although plenty of technical hurdles exist as well.

Unfortunately, software security might be new to your organization's appointed "application security" czar—if one even exists. Even knowing where to start often proves a serious challenge. The first step toward establishing an enterprise-wide software security initiative is to assess the organization's current software development and security strengths and weaknesses. Yes, this applies to software built in-house, outsourced, purchased off-the-shelf, or integrated as part of a vendor "solution."

As an exercise, ask yourself the first question in my imaginary software security assessment: "How much software does my organization purchase compared to how much it builds in-house?" If the overwhelming majority of deployed software is outsourced, software security looks a lot more like outsourced assurance than it does building security in! Most organizations do quite a bit of both, so we'll have to solve both problems.

7.3.1. Common Pitfalls

Whether tackling the problem formally or informally, top-down or bottom-up, organizations hit the same roadblocks as they prepare to build and buy more secure applications. How each organization overcomes these roadblocks depends a great deal on its strengths and weaknesses: No one-size-fits-all approach exists. Just knowing some of the landmines might help you avoid them, though. Let's look at some of the most common ones.

Lack of Software Security Goals and Vision

It bears repeating: The first hurdle for software security is cultural. It's about how software resists attack, not how well you protect the environment in which the software is deployed. Organizations are beginning to absorb this concept, but they don't know exactly what to do about it. Their first reaction is usually to throw money and one of their go-getters at it. He or she might make some progress initially by defining some application security guidelines or even buying a static analysis tool—essentially, picking the low-hanging fruit.

Although it sounds compelling, avoid charging off to win this easy battle. If your organization is large, you don't need to be reminded of what role politics plays. At the director level, headcount, budget, and timelines are the system of currency, and demanding that development teams adhere to guidelines requiring development they haven't budgeted for, or imposing a tool that spits out vulnerabilities for them to fix prior to release, can quickly send software security efforts into political deficit.

To use a war analogy, each of the chief information officer's majors must understand their role in software security prior to going into battle. To win, each major will have to take on at least a small amount of responsibility for software security, but the most crucial aspect of success is for each of them to know his or her responsibility and when to collaborate.

Put simply, without executive sponsorship, a unified understanding of roles, responsibilities, and a vision for software security, the effort will sink quickly into political struggle or inaction.

Creating a New Group

Some organizations respond to the software security problem by creating a group to address it. Headcount and attention are necessary, but it's a mistake to place this headcount on an island by itself. It's an even bigger mistake to use network security folks to create a software security capability—they just don't understand software well enough.

Software security resources must be placed into development teams and seen as advocates for security, integration, and overcoming development roadblocks.

Software Security Best Practices Nonexistent

Security analysts won't be much more effective than penetration testing tools if they don't know what to look for when they analyze software architecture and code. Likewise, levying unpublished security demands on developers is nonproductive and breeds an us-versus-them conflict between developers and security.

Instead, build technology-specific prescriptive guidance for developers. If the guidance doesn't explain exactly what to do and how to do it, it's not specific enough. Specific guidance removes the guesswork from the developer's mind and solves the problem of consistency between security analysts.

Software Risk Doesn't Support Decision Making

Although most organizations view critical security risks as having the utmost importance, project managers constantly struggle to apply risk-management techniques. The first reason for this is a lack of visibility. Even if a technical vulnerability is identified, analysts often don't fully understand its probability and impact. Rarely does an organization use a risk-management framework to consistently calculate a risk's impact at the project-management or portfolio level.

Establish a common risk framework as part of governance efforts to gain business owners' understanding and respect if you want the organization to choose security risk over time-to-market or if you need additional capital when making release decisions.

Tools as the Answer

Companies often believe that an authentication, session management, data encryption, or similar product protects their software completely. Although they serve as lynchpins of an organization's software security proposition, most organizations have a weak adoption of these tools at

best. What's worse is that these technologies are often deployed without being properly vetted. Not only do the products themselves possess vulnerabilities, but the organization's development teams weren't consulted to help with deployment, making integration difficult if not infeasible. Even if adoption of these tools were complete, they would not in and of themselves assure that an application could resist attack. Too often, architecture review is reduced to a checklist: "Did you integrate with our single sign-on and directory service tools?" "Yes? Then you're done." It's no wonder these applications still possess exploitable architectural flaws.

Penetration testing and static analysis tools aren't panaceas either. These tools help people find vulnerabilities, but there's a lot more to building security into software applications than running these tools, as we'll see.

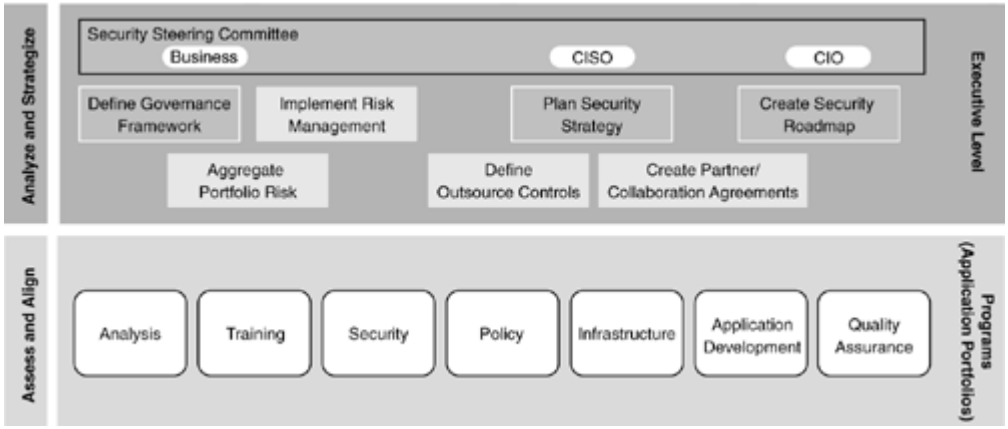
7.3.2. Framing the Solution

An enterprise software security framework (ESSF) is a new way of thinking about software security more completely at the enterprise level, targeting the problem directly without demands for massive headcount, role changes, or turning an IT shop upside down to prioritize security ahead of supporting the business that funds it. ESSFs align the necessary people, know-how, technologies, and software development activities to achieve more secure software. Because every organization possesses different strengths and weaknesses and, most important, faces different risks as a result of using software, ESSFs will differ across organizations. There are, however, certain properties that all good ESSFs will possess.

"Who, What, When" Structure

To align each group's role in achieving secure software, an organization's ESSF should possess a "who, what, when" structure—that is, the framework should describe what activities each role is responsible for and at what point the activity should be conducted. Because building security into applications requires the collaboration of a wide variety of disciplines, the framework should include roles beyond the security analysts and application development teams. **Figure 7–1** shows column headings under which an ESSF might list each role's responsibility. The boxes outlined in a lighter shade represent each role's first steps.

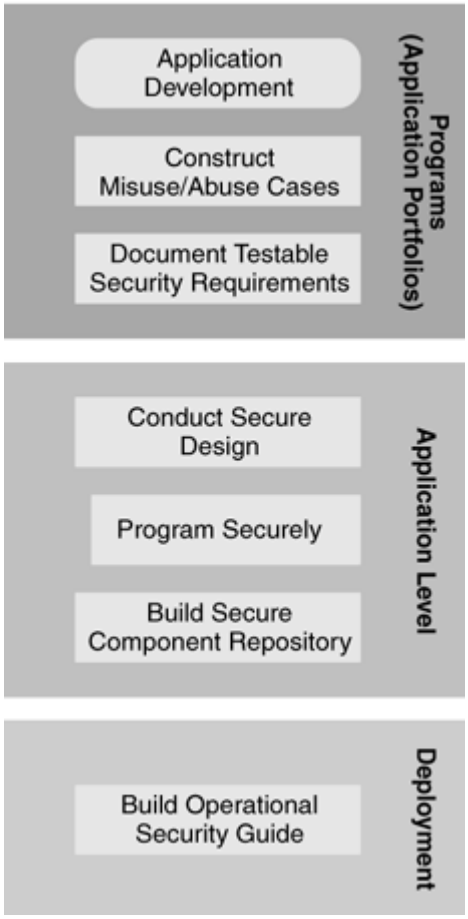
Figure 7-1. *Role responsibilities: who*



You might not recognize some of the group names in the figure. One organization's infrastructure group is another's shared services or architecture office, or something else entirely, and that's okay. Another subtlety involves reporting relationships—although they're important, don't get wound up in them when defining an ESSF. Focus on who needs to do what.

Figure 7–2 shows a partial enumeration of activities for which a particular role is responsible. Each group further defines how they accomplish each of their framework activities. For example, the business owner of development might decide to build a handbook to walk developers stepwise through the process of programming securely. It's just as likely this wouldn't get traction, though, so the ESSF could mandate training for developers before they're unleashed into the development organization.

Figure 7-2. *Role activities: what*



In this relative ordering, teams know which activities depend on others. Providing any more of a detailed "when" diagram can be seen as offensive and overly constraining to each suborganization. Let people conduct detailed project planning around these activities themselves.

It's unclear what activities will compose your organization's ESSF, but here are a few gotchas to avoid:

• Don't place technologies or products, such as "single sign-on," in the framework's boxes.

• Don't demand that teams begin conducting every activity on day one. Slowly introduce the simplest activities first, then iterate.

• Avoid activities that produce unverifiable artifacts or results, such as "collaborate with security here."

Remember, the framework's primary goal is to align people and their responsibilities, so keep the visuals about who does what activities when.

Focus on Resisting Attack, Not Including Security Features

Security is an emergent property of an application, not just an amalgam of security features. In an attempt to simplify the problem and make initial strides, organizations often get stuck in a feature-centric mode: They tell themselves, "If we just encrypt our HTTP connections and authenticate users, we're doing enough." Thinking about how to leverage the security features of toolkits, languages, and application servers within an application is good and necessary—it just isn't sufficient. To be successful, the philosophy of "resisting attack" must pervade each and every ESSF activity. Avoid the feature trap by establishing a goal of improving attack resistance in each activity from its inception. Here are some guidelines:

• Construct misuse/abuse cases.

• Model the threats each application faces.

• Assess applications against a threat model, including misuse/abuse cases.

• Train using vulnerability case studies.

• Define standards based on risk and vulnerabilities.

• Avoid relying on security-feature checklists.

• Avoid relying solely on API-guide security standards and training.

Like adopting framework activities, attempting to adhere to each of these guidelines on day one can be too onerous. Build on your organization's current strengths, infusing this guidance opportunistically.

Possess Five Competencies

Regardless of how an organization operates, every good ESSF addresses five pursuits in one form or another, as described in **Table 7–1**. Organizations should iteratively raise their competencies in each of these pursuits gradually as they adopt their ESSF.

Table 7-1. *Competencies for Effective Enterprise Software Security*

Enterprise software security framework	The ESSF defines an organization's approach to software security and describes roles, responsibilities, activities, deliverables, and measurement criteria. It also includes a communication plan for enterprise-wide rollout. Enterprise software and data architectures are essential anchors of the goal-state an ESSF defines. Definition of and migration toward a secure enterprise architecture are thus part of the framework competency.
Knowledge management, training	An organized collection of security knowledge is likely to include policy, standards, design and attack patterns, threat models, code samples, and eventually a reference architecture and secure development framework. Another element of this competency is the development and delivery of a training curriculum. Topics include security knowledge as well as help for conducting assurance activities. This pursuit also includes new courseware, along with retrofitting of existing courseware to software security concepts.
Security touchpoints	The definition of tasks and activities that augment existing development processes (formally or informally) help developers build security into any custom software development process, as well as in-place outsource assurance and commercial off-the-shelf validation processes. This competency defines how to assure software. See [McGraw 2006].
Assurance	The execution of security touchpoint activities provides assurance—conducting a software architectural risk assessment, for example, validates that security requirements were translated into aspects of the software's design and that the design resists attack. Assurance activities rely heavily on the knowledge and training competency to define what to look for. Tool adoption is likely to be part of this pursuit in the short to medium term. It will involve the purchase, customization, and rollout of static analysis tools as well as dynamic analysis aides. Your organization might have already adopted a penetration-testing product, for instance.
Governance	In the context of an ESSF, governance is competency in measuring software-induced risk and supporting an objective decision-making process for remediation and software release. This competency involves creating a seat at the project management table for software risk alongside budget and scheduling concerns. Governance should also be applied to the rollout and maturation of an organization's ESSF. The framework's owners can measure project coverage and depth of assurance activities, reported risks (and their severity), and the progress of software security knowledge and skill creation, among other things.

7.3.3. Define a Roadmap

Each competency depends somewhat on the others, and growing each effectively demands thoughtful collaboration. It's foolish to attempt to understand all the subtle interdependencies from the start and attempt a "big bang" rollout. Instead, good ESSFs leverage key initial successes in support of iterative adoption and eventual maturation. Keep two things in mind:

• *Patience.* It will take at least three to five years to create a working, evolving software security machine. Initial organization-wide successes can be shown within a year. Use that time to obtain more buy-in and a bigger budget, and target getting each pursuit into the toddler stage within the three-year timeframe.

• *Customers.* The customers are the software groups that support the organization's lines of business. Each milestone in the roadmap should represent a value provided to the development organization, not another hurdle.

Thankfully, the organizations that have been doing this work for a few years now are starting to share some of their experiences. Expert help is increasingly available, too. As always, use your community resources, and good luck being the agent of change in your organization!

7.4. How Much Security Is Enough?

Prior to selecting which security governance and management actions to take and in what order, you must answer the following question: How much security is enough? One way to tackle this question is to formulate and answer the set of security strategy questions presented in **Chapter 1** (see **Section 1.7.1**), identify a means for determining your definition of adequate or acceptable security, and use these as inputs for your security risk management framework.

7.4.1. Defining Adequate Security

Determining adequate security is largely synonymous with determining and managing risk. Where possible, an organization can implement controls that satisfy the security requirements of its critical business processes and assets. Where this is not possible, security risks to such processes and assets can be identified, mitigated, and managed at a level of residual risk that is acceptable to the organization.

Adequate security has been defined as follows: "The condition where the protection strategies for an organization's critical assets and business processes are commensurate with the organization's tolerance for risk" **[Allen 2005]**. In this definition, *protection strategies* include principles, policies, procedures, processes, practices, and performance indicators and measures—all of which are elements of an overall system of controls. [5]

An *asset* is anything of value to an organization. Assets include information such as enterprise strategies and plans, product information, and customer data; technology such as hardware, software, and IT-based services; supporting facilities and utilities; and items of significant, yet largely intangible value such as brand, image, and reputation. Critical assets are those that directly affect the ability of the organization to meet its objectives and fulfill its critical success factors **[Caralli 2004a]**. The extent to which software is the means by which digital assets are created, accessed, stored, and transmitted provides one compelling argument for ensuring that such software has been developed with security in mind.

A *process* is a series of progressive and interdependent actions or steps by which a defined end result is obtained. Business processes create the products and services that an organization offers and can include customer relationship management, financial management and reporting, and management of relationships and contractual agreements with partners, suppliers, and contractors.

Risk Tolerance

An organization's tolerance for risk can be defined as "the amount of risk, on a broad level, an entity is willing to accept in pursuit of value (and its mission)" **[COSO 2004]**. Risk tolerance influences business culture, oper-

ating style, strategies, resource allocation, and infrastructure. It is not a constant, however, but rather is influenced by and must adapt to changes in the environment.

Defining the organization's tolerance for risk is an executive responsibility. Risk tolerance can be expressed as impact (potential consequences of a risk-based event), likelihood of a risk's occurrence, and associated mitigating actions. For identified and evaluated risks, it could be defined as the residual risk the organization is willing to accept after implementing risk-mitigation and monitoring processes **[Allen 2005]**.

Risk tolerance can be expressed both qualitatively and quantitatively. For example, we might define high, medium, and low levels of residual risk. An example is a policy to take explicit and prioritized action for high- and medium-level risks and to accept (monitor) low-level risks as the default condition.

With the benefit of this description, a useful way to address the question "How much security is enough?" is to first ask, "What is our definition of adequate security?" To do so, we can explore the following more detailed questions:

• What are the critical assets and business processes that support achieving our organizational goals? What are the organization's risk tolerances, both in general and with respect to critical assets and processes?

• Under which conditions and with what likelihood are assets and processes at risk? What are the possible adverse consequences if a risk is realized? Do these risks fit within our risk tolerances?

• In cases where risks go beyond these thresholds, which mitigating actions do we need to take and with which priority? Are we making conscious decisions to accept levels of risk exposure and then effectively managing residual risk? Have we considered mechanisms for sharing potential risk impact (e.g., through insurance or with third parties)?

• For those risks we are unwilling or unable to accept, which protection strategies do we need to put in place? What is the cost–benefit relation-

ship or return on investment of deploying these strategies?

• How well are we managing our security state today? How confident are we that our protection strategies will sustain an acceptable level of security 30 days, 6 months, and 1 year from now? Are we updating our understanding and definition of our security state as part of normal planning and review processes?

7.4.2. A Risk Management Framework for Software Security[6]

As introduced in **Chapter 1**, a necessary part of any approach to ensuring adequate security (including an adequate level of software security) is the definition and use of a continuous risk management process. Software security risks include risks found in the outputs and results produced by each life-cycle phase during assurance activities, risks introduced by insufficient processes, and personnel-related risks. The risk management framework (RMF) described here can be used to implement a high-level, consistent, iterative risk analysis that is deeply integrated throughout the SDLC.

Five Stages of Activity

Figure 7–3 shows the RMF as a closed-loop process with five fundamental activity stages:

1. Understand the business context.

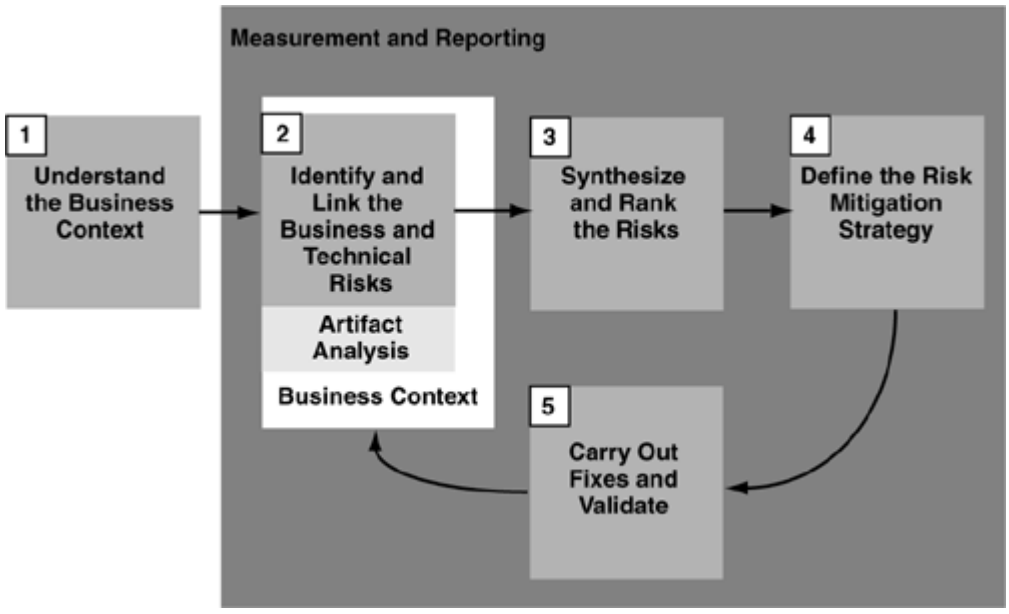
2. Identify the business and technical risks.

3. Synthesize and prioritize the risks, producing a ranked set.

4. Define the risk mitigation strategy.

5. Carry out required fixes and validate that they are correct.

Figure 7-3. *A software security risk management framework*



Each of these stages is briefly summarized below. Critical business decisions, including release readiness, can be made in a more straightforward and informed manner by identifying, tracking, and managing software risks explicitly as described in the RMF.

1. Understand the Business Context

Software risk management occurs in a business context. The increasing integration of business processes and IT systems means that software risks often have serious and specific implications for the organization's mission. Given that resources are rarely unlimited, mitigation of software risks can and should be prioritized according to the severity of the related business risks.

Central to the notion of risk management is the idea of describing impact. Without a clear and compelling tie to either business or mission consequences, technical risks, software defects, and the like are not often compelling enough on their own to warrant action. Unless software risks are described in terms that business people and decision makers understand, they will likely not be addressed.

Risks are unavoidable and are a necessary part of software development. Management of risks, including the notions of risk aversion and technical tradeoff, is deeply affected by the relevant business motivation. Thus the first stage of software risk management involves getting a handle on the business situation. Commonly, business goals are neither obvious nor ex-

plicitly stated. In some cases, the risk analyst may even have difficulty expressing these goals clearly and consistently.

During this stage, the analyst must extract and describe business goals, priorities, and circumstances to understand which kinds of software risks are important to care about and which business goals are paramount. Business goals may include, for example, increasing revenue, meeting service-level agreements, reducing development costs, and generating a high return on investment.

2. Identify Business and Technical Risks

Business risks directly threaten one or more of a customer's business goals. The identification of such risks helps to clarify and quantify the possibility that certain events will directly affect business goals. Business risks have impacts that include direct financial loss, damage to brand or reputation, violation of customer or regulatory constraints, exposure to liability, and increased development costs. The severity of a business risk should be expressed in terms of financial or project management metrics. These parameters may include, for example, market share (percentage), direct cost, level of productivity, and cost of rework.

The process of business risk identification helps to define and guide the use of particular technical methods for extracting, measuring, and mitigating software risks for various software artifacts such as requirements, architecture, and design specifications. The identification of business risks provides a necessary foundation that allows software risks (especially their impacts) to be quantified and described in business terms.

Central to this stage of the RMF is the ability to discover and describe technical risks and map them (through business risks) to business goals. A technical risk is a situation that runs counter to the planned design or implementation of the system under consideration. For example, this kind of risk may give rise to the system behaving in an unexpected way, violating its own design constraints, or failing to perform as required. Technical risks can also be related to the process used to develop software—that is, the process an organization follows may offer opportunities for mistakes in design or implementation. Technical risks involve im-

pacts such as unexpected system crashes, absence or avoidance of controls (audit or otherwise), unauthorized data modification or disclosure, and needless rework of artifacts during development.

Technical risk identification is supported by the practices described in **Chapter 4**, Secure Software Architecture and Design, which discuss the identification, assessment, prioritization, mitigation, and validation of the risks associated with architectural flaws.

3. Synthesize and Prioritize Risks

Large numbers of risks inevitably become apparent in almost any system. Identifying these risks is important, but it is the prioritization of these risks that leads directly to creation of value. Through the activities of synthesizing and prioritizing risks, the critical "Who cares?" question can (and must) be answered. Synthesis and prioritization should answer questions such as "What shall we do first, given the current risk situation?" and "What is the best allocation of resources, especially in terms of risk mitigation activities?" The prioritization process must take into account which business goals are the most important to the organization, which goals are immediately threatened, and how risks that are likely to be realized may affect the business. The output of this stage is a list of all the risks along with their relative priorities for resolution. Typical risk metrics might include, for example, risk likelihood, risk impact, risk severity, and number of risks emerging and mitigated over time.

4. Define the Risk Mitigation Strategy

Given a set of prioritized risks from stage 3, stage 4 creates a coherent strategy for mitigating the highest-priority risks in a cost-effective manner. Any suggested mitigation activities must take into account cost, time to implement, likelihood of success, completeness, and impact over the entire set of risks. A risk mitigation strategy must be constrained by the business context and should consider what the organization can afford, integrate, and understand. The strategy must also specifically identify validation techniques that can be used to demonstrate that risks are properly mitigated. Typical metrics to consider in this stage are financial in nature and include, for example, estimated cost of mitigation actions, re-

turn on investment, method effectiveness in terms of dollar impact, and percentage of risks covered by mitigating actions. Typically, it is not cost-effective to mitigate all possible risks, so some level of residual risk will remain once mitigation actions are taken. Of course, these residual risks need to be regularly reviewed and consciously managed.

5. Fix the Problems and Validate the Fixes

Once a mitigation strategy has been defined, it must be executed. Artifacts in which problems have been identified (such as architectural flaws in a design, requirements collisions, or problems in testing) should be fixed. Risk mitigation is carried out according to the strategy defined in stage 4. Progress at this stage should be measured in terms of completeness against the risk mitigation strategy. Good metrics include, for example, progress against risks, open risks remaining, and any artifact quality metrics previously identified.

This stage also involves application of previously identified validation techniques. The validation stage provides some confidence that risks have been properly mitigated through artifact improvement and that the risk mitigation strategy is working. Testing can be used to demonstrate and measure the effectiveness of risk mitigation activities. The central concern at this stage is to confirm that software artifacts and processes no longer hold unacceptable risks. This stage should define and leave in place a repeatable, measurable, verifiable validation process that can be run from time to time to continually verify artifact quality. Typical metrics employed during this stage include artifact quality metrics as well as levels of risk mitigation effectiveness.

Measurement and Reporting on Risk

The importance of identifying, tracking, storing, measuring, and reporting software risk information cannot be overemphasized. Successful use of the RMF depends on continuous and consistent identification, review, and documentation of risk information as it changes over time. A master list of risks should be maintained during all stages of RMF execution and continually revisited, with measurements against this master list being regularly reported. For example, the number of risks identified in various

software artifacts and/or software life-cycle phases can be used to identify problem areas in the software process. Likewise, the number of risks mitigated over time can be used to show concrete progress as risk mitigation activities unfold.

As you converge on and describe software risk management activities in a consistent manner, you'll find that the basis for measurement and common metrics emerges (see **Section 7.5.6**). Such metrics should help your organization achieve the following ends:

• Better manage business and technical risks, given particular quality goals

• Make more informed, objective business decisions regarding software (such as whether an application is ready to release)

• Improve internal software development processes and thereby better manage software risks

The Multilevel-Loop Nature of the RMF

The RMF shown in **Figure 7–3** has a clear loop (a single pass through the stages) that depicts risk management as a continuous and iterative process. Although the five stages are shown in a particular order in **Figure 7–3**, they may need to be applied over and over again throughout a project, and the order of stage execution may be interleaved.

There are two main reasons for this complication. First, risks can crop up at any time during the software life cycle. One natural way to apply a cycle of the loop is during each software life-cycle phase. For example, software risks should be identified, ranked, and mitigated (one loop) during requirements and again during design (another loop). Second, risks can crop up between stages, regardless of where the software is in its development life cycle or in its development process.

A further complication is that the RMF process can be applied at several levels of abstraction. The top level is the project level, meaning that each stage of the loop clearly must have some representation for an entire

project so that risks can be effectively managed and communicated by the project manager. Next comes the software life-cycle phase level: Each stage most likely has a representation for the requirements phase, the design phase, the architecture phase, the test planning phase, and so on. A third level is the artifact level. Each stage has a representation during both requirements analysis and use-case analysis, for example. Fortunately, a generic description of the validation loop is sufficient to capture critical aspects at all of these levels at once.

The risk management process is, by its very nature, cumulative and sometimes arbitrary and difficult to predict (depending on project circumstances). Specific RMF stages, tasks, and methods (described serially here) may occur independently of one another, in parallel, repeatedly, and somewhat randomly as new risks arise.

To summarize, the level of adequate security as defined here is constantly changing in response to business and risk environments and variations in the level of risk tolerance that management is willing to accept. Effectively achieving and sustaining adequate security and the use of a risk management framework demands that this work be viewed as a continuous process, not a final outcome. As a result, processes to plan for, monitor, review, report, and update an organization's security state must be part of normal day-to-day business conduct, risk management, and governance, rather than simply a one-shot occurrence.

In addition to the sources cited here, refer to "Risk-Centered Practices" **[BSI 34]** and *Software Security: Building Security In* **[McGraw 2006]** for further implementation details.

7.5. Security and Project Management[7]

This section describes how security influences project plans and management actions and suggests several approaches for inserting security practices into a defined SDLC as described in previous chapters. Continuous risk management and periodic risk assessment are key activities that help

guide project managers in determining which security practices to incorporate in each life-cycle activity and to what degree.

Software security requirements affect project planning and monitoring, specifically with respect to the following aspects of the project:

• The project's scope

• The project plan, including the project life cycle, which reflects software security practices

• Tools, knowledge, and expertise

• Estimating the nature and duration of required resources

• Project and product risks

7.5.1. Project Scope

Security's impact on the scope of the project has several dimensions that need to be considered throughout project planning and execution. These dimensions influence all SDLC activities and need to be specifically addressed in the final software and system before they are approved for release:

• The type and number of threats A risk assessment (as described in **Section 7.4.2**) can help in identifying the highest-priority threats and the profiles of the most likely attackers.

• The sophistication of and resources available to the attacker Straightforward preventive measures may offer sufficient protection from the inexperienced attacker. By contrast, protecting against experienced external attackers, those with substantial resources, and "insiders" will require more elaborate tactics.

• The desired response to an attack A *passive response* does not depend on the system having knowledge of an attack and is typically preventive in nature. For example, input validation is a passive response that prevents a significant number of attacks. An *active response* is an action that takes

place when a fault is detected. For example, an active response that improves reliability in the event of a hardware failure might be automatic failover of processing to a redundant system. A simple active response might be an automatic system shutdown when an attack is detected to protect resources, but a more frequently preferred objective for an active response is to continue to provide essential services during an attack by dynamically changing system behavior. Hence, an active response typically increases software complexity.

• The level of required assurance that the system meets its security requirements In practice, the assurance level depends on the consequences of a security failure. Security governance is typically associated with systems that require medium or high assurance. High-assurance systems include those that are important for national defense and for domains such as health care and nuclear power. Medium-assurance systems are those for which the consequences of a risk could reasonably lead to substantial reduction in shareholder value, the leakage of confidential business information, legal liability above normal business liability insurance, or substantial civil action or negative publicity.[8] Medium assurance could, for example, be applicable to corporate financial systems, manufacturing control systems, and the information systems used for critical infrastructure services such as power and water.[8] Cohen, Fred. Burton Group presentation at Catalyst 2005. Access to sensitive information may have to satisfy legal, regulatory, or fiduciary duties; contractual obligations; or voluntary requirements such as the protection of proprietary data. Those requirements raise the importance of security governance. In particular, regulatory compliance may depend on formalizing governance and risk management and, for each regulation, may require specifying the scope in terms of the responsibilities and roles for personnel and IT systems.

7.5.2. Project Plan

The nature of security risks and their consequences affect both project planning and resources. Actions to mitigate low-consequence and low-likelihood risks can often be left to the discretion of the project leader with limited management review. Conversely, the management of high-probability risks with medium-level consequences would likely require expert assistance and a well-defined, systematic review process.

All too often, software errors that render systems vulnerable to cyber-attack are introduced as a result of disconnects and miscommunications during the planning, development, testing, and maintenance of any system or software component. For example:

1. The complexity associated with product development may be a consequence of tight component integration to meet market demands for functionality or performance. Products typically have extensibility requirements so that they can be tailored for a specific customer's operating environment. The complexity induced by those product requirements also increases the risk that those features might be exploited.

2. Shared services typically aggregate risks. A failure in shared software or infrastructure services could affect multiple systems. The level of software assurance required for the shared components should be higher than that required than for the systems in which they are deployed. The higher assurance and aggregation of risks implies that the risks for shared services should include the full spectrum of integrity, confidentiality, and availability issues.

3. System integration has to resolve any mismatches with both internal and outsourced development. One mechanism to encourage better integration might be to specify the software assurance criteria for each component, such as completed code analysis for all delivered software. There will probably be differences in the software assurance requirements among components developed in-house and those commercially available.

Given these concerns, communication and links among life-cycle activities, among multiple development teams, and between the system development and operational use environments need to be addressed by project managers.

Software Security Practices in the Development Life Cycle

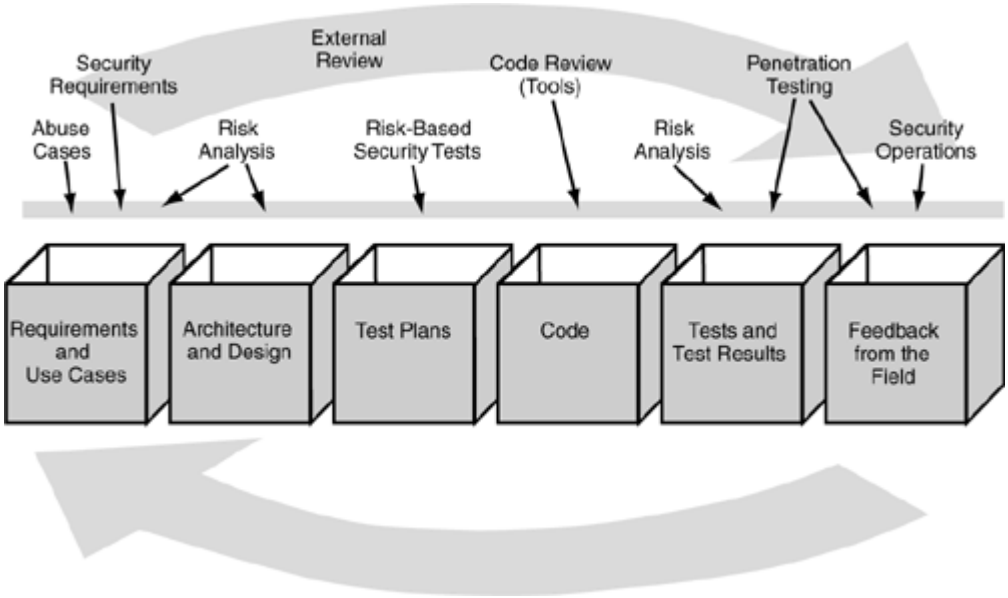
As stated earlier, a necessary part of any project planning effort that includes requirements for software security is the use of a continuous risk management process that includes risk assessment. The factors involved

with a risk assessment that is done early in the software development process are predominantly business oriented rather than technical. You'll want to ensure that business-knowledgeable stakeholders participate in risk assessment and analysis.

Architectural risk analysis is an example of an important software security practice. The software architecture describes the system structure in terms of components and specified interactions. The increased system specificity provided by this architecture calls for a more detailed description of security threats and desired system responses to them. Thus an architectural risk assessment can review the threats, analyze how the architecture responds to them, and identify additional risks introduced by the architecture. For example, attack patterns would be rather abstract for a preliminary risk assessment, but would become more detailed as the software architecture and detailed design evolve. (See **Section 7.4.2** and **Section 4.2**. Also see **Chapter 6** for a more detailed discussion of system development and integration issues, along with recommended mitigations.)

As introduced in **Chapter 1**, **Figure 7–4** depicts one example of how to incorporate security into the SDLC using the concept of touchpoints **[McGraw 2006**; **Taylor 2005]**. Software security best practices (touchpoints shown as arrows) are applied to a set of software artifacts that are created during the software development process (the boxes). While **Figure 7–4** may convey a traditional waterfall development approach, most of today's organizations actually use an iterative approach and thus cycle through the touchpoints more than once as the software evolves.

Figure 7-4. *Software development life cycle with defined security touchpoints* **[McGraw 2006]**



The Security Development Lifecycle—SDL: A Process for Developing Demonstrably More Secure Software **[Howard 2006]** provides an additional example of a pragmatic way to address security during development; it is being used successfully at Microsoft. CLASP (Comprehensive, Lightweight Application Security Process) **[BSI 12]**, as was mentioned in **Chapter 3**, is an activity-driven, role-based set of process components guided by formalized best practices. CLASP is designed to help software development teams build security into the early stages of existing and new-start SDLCs in a structured, repeatable, and measurable way. Descriptions of additional secure SDLC processes can be found in *Secure Software Development Life Cycle Processes* **[BSI 46**; **Davis 2005b]**.

The objective of including security in a defined SDLC is not to overhaul an existing process totally, but rather to add well-defined security practices and security deliverables. The implementation of these practices depends on the characteristics of the software. For example, risk analysis and assessment for an integrated system has different requirements than the risk assessment of a commercial product or an infrastructure component. The differences in software security issues and project management actions among products, application and integrated systems, and systems of systems are discussed in more detail in an article titled "The Influence of System Properties on Software Assurance and Project Management" on the BSI Web site **[BSI 36]**.

Activities Required to Complete Deliverables

Regulatory or contractual compliance may require demonstrating that the software provides the necessary controls when accessing sensitive information (that is, the production of an assurance case—see **Section 2.4**). Meeting security compliance requirements typically increases the software's complexity. For example, business process compliance may require showing that the composition and interactions of multiple applications maintain the required controls and feedback.

Delivering "secure" software requires demonstrating that the desired level of assurance has been achieved. While demonstrating that a system provides the required functionality is an essential aspect of software assurance, software security assurance depends more on demonstrating what a system does *not* do. Does improper input lead to a system failure or enable an attacker to bypass authentication or authorization defenses? (See also **Section 5.4**.) The production of such an assurance case must be planned and managed. An assurance case provides an argument for how the software addresses an identified risk. That argument typically is based on assumptions about how the software behaves under certain operating conditions. Hence, an early step in building an assurance case is to provide evidence that the software behavior satisfies the assumptions of the assurance argument. Note that the production of an assurance case is an incremental activity: The assurance case should evolve to describe how the architecture contributes to meeting security requirements, and the architectural risk assessment and analysis should provide evidence that the architecture satisfies those requirements.

An assurance case may be part of the requirements for contracted development to address such questions as "How will the assurance of delivered software be demonstrated?" and "Do the assurance cases for the supplied software support the assurance argument for the integrated system?"

7.5.3. Resources

Tools

The software development environment should be at least as secure as the planned security level of the software being produced. Appropriate controls for and configuration management of development artifacts are

essential. As part of developing those controls, specific tools may be required to aid in the production or testing of secure software, such as for static code analysis.

The security functionality for authentication, authorization, and encryption is typically composed of commercially supplied components that can be tailored for a specific operational environment. These components must have the required assurance level.

As assurance levels rise, the development process should provide the necessary control and information protection mechanisms. Change management must be conducted according to a defined, repeatable process, with zero tolerance for unauthorized changes. (See the article titled "Prioritizing IT Controls for Effective, Measurable Security" on the BSI Web site **[BSI 37]**.) High-assurance configuration management must support requirements for audit, traceability, and process enforcement. For very sensitive code segments, security governance may require that changes always be made by two developers to limit the ability of an individual to insert malicious code.

Knowledge and Expertise

Security expertise on most projects is limited and may be provided via an internal means or a contracted service. Determining how best to allocate this limited resource is challenging even when security activity involves only networks, authentication, and access control. When security has to be incorporated into application development, this expertise is even more difficult to come by. Also, any increase in the level of assurance can significantly affect the need for security and software engineering expertise.

The security expertise required to develop more secure software can be classified into two categories:

• Knowledge of security functionality and features, such as the specification and implementation of access control, authentication, and encryption. Security functionality specialists should be aware of the security issues associated with development and project management.

• The skills to identify and mitigate exploitable vulnerabilities.

Unfortunately, software development teams rarely have the necessary security expertise needed to satisfy this need. Vulnerabilities may reside in the least exercised parts of the system or depend on aspects of system interfaces that are highly unlikely to occur or difficult to predict. Software development teams may miss these types of vulnerabilities because they normally concentrate on the core software and security functionality.

Project managers need to ensure that both types of expertise are available to their development teams throughout the SDLC (see sidebar).

Tasks such as risk assessment, architectural risk analysis, and code review require significant security expertise. Other software security practices can be implemented with somewhat less experience. For example, although extensive security knowledge may be necessary to configure a tool for the static analysis of source code, the use of such a tool may not require the same level of expertise. (See **Section 5.2**.)

Testing provides a second example. Penetration testing is often part of an acceptance test or certification process. Penetration testing might be implemented by a **red team**—that is, security experts who attempt to breach the system defenses. Fuzz testing is a simple form of penetration testing that finds software defects by purposely feeding invalid and ill-formed data as input to program interfaces **[Arkin 2005**; **Howard 2006]**. Fuzz testing does not replace the need for testing that targets explicit security risks, but it is an approach that can be used without detailed knowledge of security vulnerabilities. (See **[BSI 23]** for a discussion of the effective use of fuzz testing.)

7.5.4. Estimating the Nature and Duration of Required Resources

An increase in the required assurance level can have a significant impact on project cost and schedule, and additional development skills, development practices, and tool support will be required to demonstrate that the desired assurance is in place. Traditional cost-saving strategies such as reusing existing components or general-purpose commercial components may not be useful when developing medium- and high-assurance systems. In addition, early estimates for staff effort and schedule are not

very reliable until a more detailed description of the software is available, such as that provided by the software architecture and detailed design, along with a more detailed model of attacker actions and possible responses.

Deployment of Security Expertise at Microsoft

Microsoft's experience with the implementation of its Security Development Lifecycle suggests that someone with security expertise must be available for frequent interactions with the development team during software design and development. A similar recommendation has been made for projects utilizing agile development **[Wäyrynen 2004]**.

Microsoft created a central security group that drives the development and evolution of security best practices and process improvements, serves as a source of expertise for the organization as a whole, and performs a final security review before software is released. For example, during the requirements phase, the product team requests the assignment of a security advisor from the central group who serves as point of contact, resource, and guide as planning proceeds. This security advisor helps the product team by reviewing plans and ensuring that the central security team plans for and identifies appropriate resources to support the product team's schedule. The security advisor also makes recommendations to the product team on security milestones and exit criteria based on project size, complexity, and risk.

See "Lessons Learned from Five Years of Building More Secure Software" **[Howard 2007]** for more details on this initiative.

Using shared services and a shared IT infrastructure across a number of application development projects can reduce component development costs but typically aggregates risks across all uses. In other words, if a shared service includes a certain risk, this risk needs to be managed by every project that uses the service. Examples of shared services that might be used by multiple applications include Web portal interfaces, encryption and public key infrastructure, content management, access con-

trol, and authentication. Project estimates need to consider and reflect the increased assurance that will need to be applied to any shared services.

The nature of the required security expertise varies over the development life cycle. General security expertise might be stretched thin in the initial planning and requirements phases, when teams without that experience require the most assistance. Security test planning should start after the architecture is defined. Although risk analysis has to be a continuing activity, the specific expertise required to perform such analysis may vary. The analysis of a detailed design may require in-depth knowledge of a specific technology, while the analysis of an implementation draws on a detailed knowledge of known exploits.

7.5.5. Project and Product Risks

If you are using relatively new protocols such as those for Web services, you may find them to be a moving target, as they continue to change to reflect the experiences of early adopters. Best practices in this context have short lives, and the lack of well-defined and proven practices adversely affects planning. Given this caveat, you might want to include a prototype or use an iterative or incremental approach.

Potential requirements for secure data access during development, secure facilities, or demonstration of capability can add great complexity and schedule concerns to projects.

Software vulnerabilities may be intentionally inserted during in-house or contracted development. These vulnerabilities can be much more difficult to find than those resulting from inadequate software security practices. Change and configuration management procedures provide some assurance for internal development.

Some security risks are inherent in the operational environment or with the desired functionality and hence are unavoidable. For example, it may be very difficult to block a well-resourced denial-of-service attack. Other risks may arise because of tradeoffs made elsewhere in the project. For example, an organization might permit employees to access information

assets using personal laptops or PDAs because the need for such access outweighs its perceived risks.

Mitigations May Create New Risks

Security mechanisms that mitigate a specific risk may create additional ones. For example, security requirements for managing identity for a large distributed system might be met by implementing authentication and authorization as infrastructure services shared by all applications. As noted earlier, the aggregation of authentication and authorization mechanisms into a shared service makes that service a single point of failure and a possible attack target. Such design decisions should involve a risk assessment to identify any new risks that require mediation, as well as the analysis of the operational costs after the system is deployed.

In summary, the requirement for secure systems and software affects many of the "knowledge areas" of project management—specifically, scoping, human resources, communications, risk management, procurement, quality, and integration. Activities such as an architectural risk assessment, threat analysis, and static analysis for the source code provide practices for specific development phases. Development controls and change management are essential development tools. However, the software assurance issues that arise during development are dynamic, meaning that project managers must maintain links between business and technical perspectives, among life-cycle phases, and among development teams. The production of an assurance case can serve as an integrating mechanism by identifying threats and desired responses and then tracing and refining the threats and responses during development.

Providing the necessary level of security assurance requires more than the development of what is typically thought of as a security architecture—that is, perimeter defenses (firewalls), proxies, authentication, and access controls. Software security can be achieved only by integrating software assurance practices into development processes. Such integration happens as an act of project management.

7.5.6. Measuring Software Security[9]

Measurement of both product and development processes has long been recognized as a critical activity for successful software development. Good measurement practices and data enable realistic project planning, timely monitoring of project progress and status, accurate identification of project risks, and effective process improvement. Appropriate measures and indicators of software artifacts such as requirements, designs, and source code can be analyzed to diagnose problems and identify solutions during project execution and reduce defects, rework (i.e., effort, resources), and cycle time. Unfortunately, useful measurements for software that have been developed to meet security requirements are still in their infancy, and no consensus exists as to which measures constitute best practices. Nonetheless, some measures and practices used in software development can be fruitfully extended to address security requirements.

Effective use of a software development measurement process for security relies first on agreeing on the desired security characteristics and measurement objectives, which can be applied to both the product and the development process. These objectives rely on having explicit system requirements—which means that security aspects must be specified early in the SDLC (see **Chapter 3**). The organization should assess the risk environment to address probable risks and translate these concerns into specific security requirements, and then design and implement a development process that ensures such requirements are built in.

Measurement objectives can be formulated that will provide insight into the software's security state or condition. Following are some examples of analytical questions that can lead to measurement objectives:

• Which vulnerabilities have been detected in our products? Are our current development practices adequate to prevent the recurrence of the vulnerabilities?

• Which process steps or activities are most likely to introduce security-related risks?

• What proportion of defects relates to security concerns and requirements? Do defect classification schemes include security categories?

• To what extent do developers comply with security-related processes and practices?

• To what extent are security concerns addressed in intermediate work products (e.g., requirements, architecture and design descriptions, test plans)? Have measures associated with security requirements and their implementation been defined and planned?

• What are the critical and most vulnerable software components? Have vulnerabilities been identified and addressed?

Architectural risk analysis (which helps identify probable types and sources of attacks) can provide significant guidance for the development processes for secure products (see **Chapter 4**). A thesis by Stuart E. Schechter at Harvard University's Department of Computer Science uses economic models for valuing the discovery of vulnerabilities in the final or end product during development **[Schechter 2004]**. His measurement of security strength depends most on threat scenarios to assign values to vulnerabilities. Many risk and threat methodologies are publicly available, including **[McGraw 2006]** and CLASP **[BSI 12]**. In addition, Microsoft has published extensive materials that delineate its approach to analyzing and mitigating threat risks during the SDLC **[Howard 2006]**.

Process Measures for Secure Development

Process artifacts that implement security measurement objectives for the development process should address the following issues:

• The presence of security policies applicable to the SDLC (e.g., roles and responsibilities, management, procedures, coding rules, acceptance/release criteria)

• Policy compliance

• The efficiency and effectiveness of the policies over time

The security measurement objectives for the development process are identical to the general measurement objectives—and they need to be included in the process implementation. Such measures could be implemented as part of an organization's integrated quality assurance function.

Although targeted for systems development and risk assessment as a whole, the NIST Special Publication *Security Metrics Guide for Information Technology Systems* **[Swanson 2003]** provides useful guidance for measurements of this type. Risk management can encompass secure coding and provides a familiar framework in which to incorporate new practices and procedures to address software security issues (see also **Section 7.4.2**).

Defect *density* is a commonly used measure of product quality. It is often computed as the number of defects discovered during system testing or during the first six months of operational use divided by the size of the system. Estimates of defects remaining in the product (calculated by techniques such as phase containment, defect depletion, and capture–recapture techniques) form a natural analogue to estimate security vulnerabilities remaining in the software. *Phase containment* of defects is an analytical technique that measures the proportion of defects originating in a phase that are detected within that same phase; it provides a good characterization of the ability of the development process to maintain quality throughout the SDLC. Refer to "Team Software Process for Secure Systems Development" [in **BSI 46**; Over 2002] for additional information on process measures for secure development.

Product Measures for Secure Development

In the product context, security concerns addressed by measurement objectives may take any of the following forms:

• Security requirements, which are based on privacy policies, legal implications, risks identified by threat assessments, and other sources, and can

be specified as to extent and completeness

• Security architecture, which reflects the specified security requirements

• Secure design criteria, where security requirements can be traced

• Secure coding practices, where integrity can be assessed and measured

Not all measures need to be complicated. Measures should be as simple as possible while still meeting the project's information needs. For example, in the requirements phase it is useful to know whether security-related concerns have been considered in specifying system requirements. This information could be summarized initially as yes or no. As experience with the measure accrues over time, however, the measure could evolve to characterize the extent to which requirements have been checked and tested against security concerns. Tools, inspections, and reviews can be used to determine the extent to which security measurement objectives are implemented during the design and coding phases.

Inspection measurements often take the form of traditional defect identification checklists, to which security-oriented items have been added. For example, you could track the percentage of sources of input that have validation checks and associated error handling. Check each input source for length, format, and type, and its associated exit flows—either (1) accepted and then executed or (2) recognized as an error or exception and not executed. The target for this measure would be 100 percent, unless performance is unacceptable as a result or this scheme costs too much to implement. Note that while this simple measure represents an improvement over no measurement for this type of vulnerability, it does not address the potentially complex issue of determining the effectiveness of an input validation technique as implemented and whether any particular datum should be counted in the tally. Resolving this issue requires ongoing tracking of this measure's performance to characterize the effectiveness of the input validation techniques used. Over time, you can benchmark these kinds of measures as performance standards.

Simple measures of enumeration and appropriate security handling for vulnerabilities provide insight into the security status of the software

during development. For example, a useful list of "Measurable Security Entities" and "Measurable Concepts" has been published by Practical Software and Systems Measurement **[PSM 2005]**. Questions generated by the PSM/DHS Measurement Technical Working Group address many of the previously mentioned issues and can provide starting points for developing measurement objectives. These questions—which form a solid basis for measurement in most development organizations, regardless of size or methods employed—can be found on the BSI Web site in the article titled "Measures and Measurement for Secure Software Development" **[BSI 38]**. In addition, a useful description of software security metrics for Web applications is provided in "A Metrics Framework to Drive Application Security Improvement" **[Nichols 2007]**.

This section has described a range of topics that project managers need to pay particular attention to when developing secure software. *Software Project Management for Software Assurance* **[Fedchak 2007]** is another comprehensive source that presents information on how software assurance and software security affect project management practices, including risk management, size and cost estimation, metrics, quality assurance, and management practices by life-cycle phase.

We close this chapter by highlighting a number of observed shifts and trends, along with supporting evidence, that describe the state of the practice in various communities and market sectors with respect to governing and managing information and software security. This information can be used to help formulate business-based arguments and implementation approaches for developing and deploying more secure software.

7.6. Maturity of Practice

Security's emergence as a governance and management concern is primarily taking place in the parts of the organization that provide and use IT. We currently see minimal attention paid to this topic during the early life-cycle phases of software and system development, but increasing attention being devoted to it during detailed design, coding, and testing.

Treating security as a governance and management concern, as a risk management concern, and as a project management concern at the earliest phases of the life cycle will likely produce more robust, less vulnerable software, thereby resulting in a decline in the reactive, fire-fighting mode now observed in most IT and system operations and maintenance organizations.

Consistent governance and management action across the organization is key. This includes attention and participation from business unit leaders, human resources, legal, audit, risk management, and finance, as well as IT and software and system development groups. This section identifies several indicators that organizations are addressing security as a governance and management concern, at the enterprise level. It summarizes how some organizations, trade associations, and market sectors are proceeding in this area. Many of the references cited here provide more detailed implementation guidance.

7.6.1. Protecting Information

One significant shift that is causing leaders to pay increasing attention to security is the need to treat information—and particularly consumer, customer, client, and employee information—with greater care, perhaps with the same care as money. Leaders understand how their organizations' reputations may suffer if this is not done competently and breaches become public.[10] Customers expect that organizations will carefully protect their privacy and their information, and they are becoming more acutely aware of the risk of identity theft posed by unintended data disclosure. U.S. federal laws such as the Sarbanes–Oxley Act for financial reports, along with state laws such as the California Database Protection Act for consumer data, have codified these concerns. The European Union's Directive on the Protection of Personal Data[11] is even more comprehensive with respect to an organization's legal duty and ethical responsibility to protect personal information.

The credit card industry has been proactive in defining a standard for all merchants that accept and process credit card information. Through the efforts of American Express, Discover Financial Services, JCB, MasterCard Worldwide, and Visa International, the Payment Card Industry Security

Standards Council was founded and acts as the steward of the Payment Card Industry Data Security Standard **[PCI 2006]**. As stated on its Web site, "The PCI DSS is a multifaceted security standard that includes requirements for security management, policies, procedures, network architecture, software design, and other critical protective measures. This comprehensive standard is intended to help organizations proactively protect customer account data." The key requirements of DSS are that member organizations will (1) build and maintain a secure network, (2) protect cardholder data, (3) maintain a vulnerability management program, (4) implement strong access control measures, (5) regularly monitor and test networks, and (6) maintain an information security policy. An article on the BSI Web site titled "Plan, Do, Check, Act" **[BSI 39]** describes how to integrate PCI DSS requirements with other accepted security standards for sustaining software security during deployment and operations.

7.6.2. Audit's Role

As part of the U.S. Critical Infrastructure Assurance Project, the Institute of Internal Auditors (IIA) held six summit conferences in 2000 to better understand the role of governance with respect to information security management and assurance. The IIA also provided guidance in 2001 in the document titled "Information Security Governance: What Directors Need to Know." This report includes case studies from General Motors, IBM, BellSouth, Intel, Sun Microsystems, the Federal Reserve Bank in Chicago, and Home Depot. Useful questions to ask that resulted from this work are listed in "Maturity of Practice and Exemplars" on the BSI Web site **[IIA 2001**; **BSI 40]**.

The Information Systems Audit and Control Association (ISACA) and its partner organization, the IT Governance Institute (ITGI), have published extensive guidance on information technology and information security governance. Their report titled "Information Security Governance: Guidance for Boards of Directors and Executive Management" **[ITGI 2006]** addresses these questions:

1. What is information security governance?

2. Why is it important?

3. Who is responsible for it?

The same report also describes how to measure an organization's maturity level relative to information security governance.

7.6.3. Operational Resilience and Convergence

In its work with the Financial Services Technology Consortium (FSTC), CERT is examining the convergence of security, business continuity, and IT operations management given their critical roles in operational risk management.[12] The intent is "to improve the operational resiliency of the organization—the ability to adapt to a changing operational risk environment as necessary" **[Caralli 2006]**. In their technical reports *Sustaining Operational Resilience: A Process Improvement Approach to Security Management* **[Caralli 2006]** and *Introducing the CERT Resiliency Engineering Framework: Improving the Security and Sustainability Processes* **[Caralli 2007]**, the authors offer an initial process improvement framework for business continuity and security. This framework is being pilot-tested with members of the FSTC and other collaboration partners.

A number of other organizations are describing their efforts to achieve organizational resilience through the integration of business continuity, operational and technology risk management, compliance, and information security and privacy, supported by audit. These integrating activities occur across products and business lines and take into account people, business processes, infrastructure, applications, information, and facilities. Indicators of success include the following outcomes:

• Reduced risk of a business interruption

• Shorter recovery time when an interruption occurs

• Improved ability to sustain public confidence and meet customer expectations

• Increased likelihood of complying with regulatory and internal service level requirements

The Alliance for Enterprise Security Risk Management is a coalition formed by ASIS International (representing the physical security community), ISACA (representing the IT audit community), and ISSA (Information Systems Security Association, representing the information security community). It is addressing "the integration of traditional and information security functions to encourage board and senior executive level attention to critical security-related issues" **[AESRM 2005]**. In its study titled "Convergence of Enterprise Security Organizations," the Alliance quotes the ASIS definition of convergence:

> The identification of security risks and interdependencies between business functions and processes within the enterprise and the development of managed business process solutions to address those risks and interdependencies.

The report goes on to describe five imperatives driving convergence[13] and the organizational implications with supporting examples. These efforts are providing evidence of the value of addressing security as part of a broader convergence effort and in support of organizational preparedness.

7.6.4. A Legal View

The American Bar Association's Privacy and Computer Crime Committee has published a "Roadmap to an Enterprise Security Program" **[Westby 2005]**. The preface to the Roadmap states the following:

> This publication was developed by a multidisciplinary team of industry representatives, government personnel, policy specialists, attorneys, technical experts, and academicians. They came together to provide a roadmap that links the various pieces of the cyber security "puzzle" into an orderly process that conforms with global standards and best practices, helps meet compliance requirements, facilitates cooperation with law enforcement, and promotes public-private sector cooperation.

The Roadmap presents a structure that includes governance, security integration and security operations, implementation and evaluation, and

capital planning and investment controls. The steps for governance include these **[Westby 2005]**:

• Establish governance structure, exercise oversight, and develop policies.

• Inventory digital assets (networks, applications, information).

• Establish ownership of networks, applications, and information; designate security responsibilities for each.

• Determine compliance requirements with laws, regulations, guidance, standards, and agreements (privacy, security, and cybercrime).

• Conduct threat and risk assessments and security plan reviews (for internal and contractor operations). This may include certification and accreditation.

• Conduct risk management based on digital asset categorization and level of risk.

7.6.5. A Software Engineering View

An emerging body of knowledge describes aspects of how to apply governance and management thinking to the engineering and development of secure software. In addition to John Steven's article "Adopting an Enterprise Software Security Framework" provided in **Section 7.3**, several other articles on the BSI Web site that were previously published in a series in *IEEE Security & Privacy* address aspects of this issue. "Adopting a Software Security Improvement Program" **[BSI 41**; **Taylor 2005]** provides several concrete steps and a progression of phases for improvement. "Bridging the Gap Between Software Development and Information Security" **[BSI 42**; **van Wyk 2005]** describes a range of secure software development activities and practices to conduct during a software development life cycle.

Chapter 10 of **Software Security***: Building Security In* **[McGraw 2006]** elaborates on several of the *IEEE Security & Privacy* articles. It describes

elements of an enterprise software security program, addressing the following concerns:

• The business climate

• Building blocks of change, including four common pitfalls:

– Over-reliance on late-life-cycle testing

– Management without measurement

– Training without assessment

– Lack of high-level commitment (particularly relevant for governance and management)

• Building an improvement program

• Establishing a metrics program, including a three-step enterprise rollout:

– Assess and plan

– Build and pilot

– Propagate and improve

• Continuous improvement

• COTS (and existing software applications), including an enterprise information architecture

• Adopting a secure development life cycle

Part I of *The Security Development Lifecycle—SDL: A Process for Developing Demonstrably More Secure Software* **[Howard 2006]** describes the need for a Secure Development Lifecycle (SDL). According to Michael Howard and Steve Lipner, "The biggest single factor in the success of SDL is executive support." Effective commitment to an SDL includes making a

statement, being visible, providing resources, and stopping the delivery of products that do not meet their security and SDL requirements. Part II of the same book describes the 12-stage SDL.

7.6.6. Exemplars

On the BSI Web site, the article titled "Maturity of Practice and Exemplars" **[BSI 40]** provides 12 examples of principles, guidelines, frameworks, and roadmaps that can assist organizations in implementing a governance-based enterprise security program. This article summarizes the efforts of several professional associations and selected market sectors and organizations, describing how they have successfully addressed security at governance and management levels. Many use strategic questions and guiding principles as starting points. Summaries are provided for the following organizations and events:

• Aberdeen Group

• American Chemistry Counsel

• Audit Associations (Institute of Internal Auditors, Information Technology Governance Institute)

• BITS

• Corporate Governance Task Force

• Corporate Information Security Working Group

• Federal Financial Institutions Examination Council

• Health Information and Management Systems Society

• ISO/IEC 27001 and ISO/IEC 17799

• National Association of Corporate Directors

• National Institute of Standards and Technology

• Payment Card Industry Data Security Standard

• Veterans Administration Data Breach

Several of these examples and their supporting references provide sufficient detail to help you start implementing a security governance and management program.

Clearly, many sectors, organizations, and organizational functions (including risk management, IT, business continuity, audit, legal, and software development) are making progress and producing results by treating security as an enterprise issue. They are taking governance and management actions to integrate security into ongoing business councils and steering groups, decision-making processes, plans, business and development processes, and measures of success.

7.7. Summary

Regardless of the extent of security practices included as part of the SDLC, software security and system security cannot be accomplished without informed, knowledgeable, committed leaders—business leaders, project managers, and technical leaders. This chapter presented recommendations and practices to aid software project managers in the management tasks of producing more secure software as well as actions to take to engage their business leaders and senior executives at the governance level.

Key recommendations and practices are as follows:

• Recognize that being security aware and understanding the importance of addressing security during software development needs to be a cultural norm.

• Engage leaders to better appreciate and understand the characteristics and actions necessary to address security as governance and management concerns, and the consequences of not doing so.

• Establish a framework and roadmap for addressing software security as an enterprise-wide undertaking and for tackling some of the pitfalls and barriers head on.

• Identify ways to determine what constitutes adequate security practice based on risk management, established levels of risk tolerance, and risk assessment.

• Put a continuous, business-driven risk management framework in place, and assess for acceptable and unacceptable levels of risk throughout the SDLC.

• Follow the recommendations for inserting security into the SDLC as part of traditional project management activities, including the use of defined security touchpoints at each life-cycle phase.

• Include security as part of the software development measurement process, including implementing suggested process and product measures.

As the examples offered in this chapter demonstrate, security is being effectively tackled today, at the enterprise level, from several points of view, representing a growing community of practice across a wide range of roles, disciplines, and market sectors.