Software Engineering Institute

Making the Business Case for Software Assurance

Nancy R. Mead
Julia H. Allen
W. Arthur Conklin
Antonio Drommi
John Harrison
Jeff Ingalsbe
James Rainey
Dan Shoemaker

April 2009

SPECIAL REPORT
CMU/SEI-2009-SR-001

CERT Program

http://www.sei.cmu.edu

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense and the Department of Homeland Security National Cyber Security Division. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

For information about purchasing paper copies of SEI reports, please visit the publications section of our website (http://www.sei.cmu.edu/publications/).

Table of Contents

Acknowledgments **vi**

Executive Summary **vii**

Abstract **ix**

1 Introduction **1**

1.1 Audience for This Guide 2

1.2 Motivators 2

1.3 How to Use This Guide 3

2 Cost/Benefit Models Overview **4**

2.1 Traditional Cost/Benefit Models 4

2.2 Investment-Oriented Models 4

2.2.1 Total Value of Opportunity (TVO) – Gartner 4

2.2.2 Total Economic Impact (TEI) – Forrester 5

2.2.3 Rapid Economic Justification (REJ) – Microsoft 6

2.3 Cost-Oriented Models 7

2.3.1 Economic Value Added (EVA) – Stern Stewart & Co 7

2.3.2 Economic Value Sourced (EVS) – Cawly & the Meta Group 8

2.3.3 Total Cost of Ownership (TCO) – Gartner 8

2.4 Environmental/Contextual Models 9

2.4.1 Balanced Scorecard – Norton and Kaplan 9

2.4.2 Customer Index: Andersen Consulting 10

2.4.3 Information Economics (IE) – The Beta Group 11

2.4.4 IT Scorecard – Bitterman, IT Performance Management Group 11

2.5 Quantitative Estimation Models 12

2.5.1 Real Options Valuation (ROV) 12

2.5.2 Applied Information Economics (AIE) – Hubbard 13

2.5.3 COCOMO II and Security Extensions – Center for Software Engineering 14

2.6 Some Common Features 15

2.6.1 General Factors 15

2.6.2 Common Factors Across Models 15

2.7 Limitations 17

2.8 Other Approaches 17

3 Measurement **18**

3.1 Characteristics of Metrics 18

3.2 Types of Metrics 19

3.3 Specific Measurements 20

3.4 What to Measure 22

3.5 SDL Example 23

4 Risk **24**

4.1 Introduction 24

4.2 Risk Definitions 25

4.3 A Framework for Software Risk Management 25

4.3.1 Understand the Business Context 26

4.3.2 Identify the Business and Technical Risks 27

4.3.3 Synthesize and Rank (Analyze and Prioritize) Risks 27

4.3.4 Define the Risk Mitigation Strategy 28
4.3.5 Fix the Problems and Validate the Fixes 28
4.3.6 Measurement and Reporting on Risk 28
4.4 Methods for Assessing Risk 29
4.5 Identifying Risks 31
4.5.1 Assets 32
4.5.2 Threats 32
4.5.3 Vulnerabilities 33
4.5.4 Impacts to Assets 33
4.6 Analyzing Risks 34
4.6.1 Business Impact 34
4.6.2 Likelihood 35
4.6.3 Risk Valuation 35
4.7 Categorizing and Prioritizing Risks 35
4.8 Mitigating Risks 36
4.8.1 Mitigations 36
4.8.2 Residual Risk 37
4.9 Summary 37

5 Prioritization 39
5.1 Foundation and Structure 39
5.2 Using the Dashboard 42

6 Process Improvement and Secure Software 45
6.1 Ensuring a Capable Process 45
6.2 Adapting the CMMI to Secure Software Assurance 46
6.2.1 Level 1 – Initial 47
6.2.2 Level 2 – Managed 47
6.2.3 Level 3 – Defined 48
6.2.4 Level 4 – Quantitatively Managed 49
6.2.5 Level 5 – Optimizing 50
6.2.6 Implementing the Process Areas 50
6.2.7 Differences Between the CMMI and Software CMM Process Areas 50
6.3 The CMMI Appraisal Process 51
6.4 Adapting ISO 15504 to Secure Software Assurance 51
6.4.1 Assessment and the Secure Life Cycle 53
6.4.2 ISO 15504 Capability Levels 56
6.5 Adapting the ISO/IEC 21287 Standard Approach to Secure Software Assurance 57
6.6 The Business Case for Certifying Trust 58
6.6.1 Certification: Ensuring a Trusted Relationship with an Anonymous Partner 59

7 Globalization 61
7.1 Outsourcing Models 61
7.1.1 Another View of Outsourcing Options 62
7.2 Costs and Benefits of Offshoring 62
7.3 Project Management Issues 63
7.4 Location 63
7.5 Possible Tradeoffs 63

8 Organizational Development 65
8.1 Introduction: Adding a New Challenge to an Existing Problem 65
8.2 Maintaining the Minimum Organizational Capability to Ensure Secure Software 65
8.3 Learning to Discipline Cats 66

8.4 Ensuring That Everybody in the Operation Is Knowledgeable 67
8.4.1 Awareness Programs 67
8.4.2 Training Programs 68
8.4.3 Education Programs 68
8.5 Increasing Organizational Capability Through AT&E 69
8.5.1 Security Recognition 69
8.5.2 Informal Realization 69
8.5.3 Security Understanding 69
8.5.4 Deliberate Control 70
8.5.5 Continuous Adaptation 70
8.6 The Soft Side of Organizational Development 71
8.7 Some General Conclusions 71

9 Case Studies and Examples 73
9.1 Background 73
9.2 Case Studies and Examples 73
9.2.1 Case 1: Large Corporation 73
9.2.2 Case 2: SAFECode 73
9.2.3 Case 3: Microsoft 74
9.2.4 Case 4: Fortify Case Study Data 75
9.2.5 Case 5: COCOMO data 75
9.3 Conclusion 75

10 Conclusion and Recommendations 76
10.1 Getting Started 76
10.2 Conclusion 77

Appendix A: The “Security” in Software Assurance 78

Appendix B: Cost/Benefit Examples 79

Appendix C: SIDD Examples 83

Appendix D: Process Improvement Background 91

Appendix E: Improving Individual and Organizational Performance 94

Appendix F: Relevance of Social Science to the Business Case 96

Bibliography 97

List of Figures

Figure 1: A Software Security Risk Management Framework 26

Figure 2: Effect of Microsoft Security "Push" on Windows and Vista 74

Figure 3: Fortify Case Study Data 75

List of Tables

Table 1: Comparison of Cost/Benefit Models 16

Table 2: Risk-Level Matrix 35

Table 3: Risk Scale and Necessary Actions 36

Table 4: SIDD Categories and Indicators 40

Table 5: Categories of Measures for Four Perspectives of the Balanced Scorecard 79

Table 6: Sample Set of Measures for Assigning Value to Software Assurance 80

Acknowledgments

We would like to acknowledge John Bailey, our colleague on the informal Business Case Team, the authors of articles on Business Case on the Build Security In website, and the speakers and participants in our workshop “Making the Business Case for Software Assurance.” All have contributed to our thinking on the subject. We further acknowledge the sponsor of the work, Joe Jarzombek, at the National Cyber Security Division in the Department of Homeland Security; John Goodenough, for his thoughtful review; and our editor, Pamela Curtis, for her constructive editorial modifications and suggestions.

Executive Summary

As software developers and software managers, we all know that when we want to introduce new approaches in our development processes, we have to make a cost/benefit argument to our executive management to convince them that there is a business or strategic return on investment. Executives are not interested in investing in new technical approaches simply because they are innovative or exciting. The intended audience for this guide is primarily software developers and software managers with an interest in assurance and people from a security department who work with developers. The definition of software assurance used in this guide is “a level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at any time during its life cycle, and that the software functions in the intended manner” [CNSS 2006]. This definition clearly has a security focus, so when the term “software assurance” appears in this guide, it will be in the context of this definition.

In the area of software assurance we have started to see some evidence of successful economic arguments (including ROI) for security administrative operations. Initially there were only a few studies that presented evidence to support the idea that investment during software development in software security will result in commensurate benefits across the entire life cycle. This picture has improved, however, and this report provides some case studies and examples to support the cost/benefit argument.

In reading through this guide, however, it will become obvious that there is no single “best” method to make the business case for software assurance. This guide contains a variety of mechanisms, and each organization using the guide must decide on the best strategies for their situation. In Section 2 we present a number of different models for computing cost/benefit. In Section 3 we discuss measurement and the need for measurement to support cost/benefit and ROI arguments. Section 4 discusses risk. Section 5 discusses prioritization, once the risks are understood. Section 6 discusses process improvement and its relationship to software assurance and business case. Section 7 discusses the topic of offshoring and its relationship to software assurance and business case. Section 8 discusses organizational development in support of software assurance and business case. Section 9 provides case studies in support of business case, and Section 10 provides our conclusions and final recommendations.

In summary, the following steps are recommended in order to effectively make the business case for software assurance.

1. **Perform a risk assessment**. If you are going to make the business case for software assurance, you need to understand your current level of risk and prioritize the risks that you will tackle.
2. **Decide what you will measure**. If you are going to have any evidence of cost/benefit, you will need to have a way of measuring the results. This may involve use of some of the models discussed in this guide, development of your own measures of interest, or use of data that you are already collecting.
3. **Implement the approach on selected projects**. Go ahead and collect the needed data to assess whether there really is a valid cost/benefit argument to be made for software assurance. The case studies that we present are the result of such implementations.

4. **Provide feedback for improvement**. Development of a business case is never intended to be a one-time effort. If your cost/benefit experiments are successful, see how they can become part of your standard practices. Assess whether you can collect and evaluate data more efficiently. Assess whether you are collecting the right data. If your cost/benefit experiments are not successful (cost outweighs benefit), ask yourself why. Is it because software assurance is not a concern for your organization? Did you collect the wrong data? Were staff members not exposed to the needed training? Are you trying to do too much?

In order to effect the changes needed to support the software assurance business case, we recommend the following steps:

1. **Obtain executive management support**. It's almost impossible to make the changes that are needed to support the business case for software assurance without management support at some level. At a minimum, support is needed to try to improve things on a few pilot projects.
2. **Consider the environment in which you operate**. Does globalization affect you? Are there specific regulations or standards that must be considered? These questions can influence the way you tackle this problem.
3. **Provide the necessary training**. One of the significant elements of the Microsoft security "push" and other corporate programs, such as IBM's software engineering education program, is a commitment to provide the needed training. The appropriate people in the organization need to understand what it is you are trying to do, why, and how to do it.
4. **Commit to and achieve an appropriate level of software process improvement**. Regardless of the process you use, some sort of codified software development process is needed in order to provide a framework for the changes you are trying to effect.

This guide and the associated references can help you get started along this worthwhile path. This culminates a multi-year investigation of ways to make the business case for software assurance. This effort included informal and formal collaboration, a workshop on the topic, and development of this report.

"Making the Business Case for Software Assurance" is an ongoing collaborative effort within the Software Assurance Forum and Working Groups, a public-private metagroup, co-sponsored by the National Cyber Security Division of the Department of Homeland Security and organizations in the Department of Defense and the National Institute for Standards and Technology. The Software Assurance Community Resources and Information Clearinghouse website at https://buildsecurityin.us-cert.gov/swa/ provides relevant resources and information about related events.

Abstract

This report provides guidance for those who want to make the business case for building software assurance into software products during each software development life-cycle activity. The business case defends the value of making additional efforts to ensure that software has minimal security risks when it is released and shows that those efforts are most cost-effective when they are made appropriately *throughout* the development life cycle. Although there is no single model that can be recommended for making the cost/benefit argument, there are promising models and methods that can be used individually and collectively for this purpose, as well as some convincing case study data that supports the value of building software assurance into newly developed software. These are described in this report.

The report includes a discussion of the following topics as they relate to the business case for software assurance: cost/benefit models, measurement, risk, prioritization, process improvement, globalization, organizational development, and case studies. These topics were selected based on earlier studies and collaborative efforts, as well as the workshop “Making the Business Case for Software Assurance,” which was held at Carnegie Mellon University in September 2008.

1 Introduction

As software developers and software managers, we all know that when we want to introduce new approaches in our development processes, we have to make a cost/benefit argument to our executive management to convince them that there is a business or strategic return on investment. Executives are not interested in investing in new technical approaches simply because they are innovative or exciting. For profit-making organizations, we need to make a case that demonstrates we will improve market share, profit, or other business elements. For other types of organizations, we need to show that we will improve our software in a way that is important—that adds to the organization's prestige, ensures the safety of troops in the battlefield, and so on.

In the area of software assurance, particularly security, we have started to see some evidence of successful ROI or economic arguments for security administrative operations, such as maintaining current levels of patches and establishing organization entities such as Computer Security Incident Response Teams (CSIRTs) [Ruefle 2008] to support security investment [Blum 2006, Gordon 2006, Huang 2006, Nagaratnam 2005]. Initially there were only a few studies that presented evidence to support the idea that investment during software development in software security will result in commensurate savings later in the life cycle, when the software becomes operational [Soo Hoo 2001, Berinato 2002, Jaquith 2002]. This picture has improved, however. As we expected early on, Microsoft has published data reflecting the results of using their Security Development Lifecycle [Howard 2006]. Microsoft is using the level of vulnerabilities and therefore the level of patches needed as a measure of improved cost/benefit [Microsoft 2009]. The reduction in vulnerabilities and related patches in recent Microsoft product releases is remarkable.

We would also refer readers to the Business Context discussion in Chapter 2 and the Business Climate discussion in Chapter 10 of McGraw's recent book [McGraw 2006] for ideas. There has been some work on a security-oriented version of COCOMO called COSECMO [Colbert 2006]; however, the focus has been more on cost estimation than on return on investment. Reifer is also working in this area on a model called CONIPMO, which is aimed at systems engineers [Reifer 2006]. Data presented by Fortify [Meftah 2008] indicates that the cost of correcting security flaws at the requirements level is up to 100 times less than the cost of correcting security flaws in fielded software. COCOMO data suggests that the cost of fixing errors of all types at requirements time is about 20 times less than the cost of fixing errors in fielded software. Regardless of which statistic is used, there would seem to be a substantial cost savings for fixing security flaws during requirements development rather than fixing them after software is fielded. For vendors, the cost is magnified by the expense of developing and releasing patches. However, it seems clear that cost savings exist even in the case of custom software when security flaws are corrected early in the development process.

At this time there is little agreement on the right kinds of models to be used for creating a business case, and although there is now some data that supports the ROI argument for investment in software security early in software development, there is still very little published data.

Our belief is that even though they may not constitute a traditional ROI argument, the methods being used to calculate cost/benefit, whether they be reduced levels of patching in the field or reduced cost of fixing security flaws when they are found early in the life cycle, are convincing.

1.1 Audience for This Guide

The intended audience for this guide is primarily software developers and software managers with an interest in security/assurance and people from a security department who work with developers.

These software developers/managers could reside in the software vendor/supplier community or reside within an in-house development team within the consumer community. The cost/benefit analysis could be quite different between these two types of communities, but it is hoped that the information in this guide will provide useful insight for both perspectives.

Software developer/managers facing the safety-critical or national security application market will almost certainly have already invested in software assurance, as their market has security expectations with an established set of requirements. Continuous improvement is the mantra of software assurance as much as it is for quality, so their business case may be looking for efficiency savings and process improvement using the latest tools and techniques. Experienced software assurance readers will still benefit from this guide, as a wide range of cost/benefit models and supporting topics are presented which could complement their existing approach.

The case is different for software vendors facing the shrink-wrap mass consumer market. This market may expect software assurance but not expect to pay a premium for it. The business case for vendors may only support an investment in raising awareness and training together with some tool evaluation to help build up relevant skills. Or they may be looking at significant investment to reduce increasing software support costs or to extend their market into communities that expect higher levels of software assurance. There is sufficient breadth and depth in this guide to help with these two ends of the investment spectrum.

Although this guide is aimed primarily at producers of software, consumers and enterprise users of software will also find it useful to justify costs associated with meeting software assurance requirements when they come to specify and procure software.

1.2 Motivators

The commonly accepted definition of software assurance is “a level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at any time during its life cycle, and that the software functions in the intended manner” [CNSS 2006]. If the reader is a software developer/manager in the safety-critical or national security application market, they will understand exactly what this means and will understand many of the problems and have experience in many of the solutions. Readers who are not from this background may find the discussion in Appendix A helpful.

Software assurance is a national security priority [PITAC 1999]. That is due to the common-sense fact that a computer-enabled national infrastructure is only going to be as reliable as the code that underlies it [Dynes 2006, PITAC 1999]. Thus, it is easy to assume that any set of activities that increase the general level of confidence in the security and reliability of our software should be on the top of everybody’s wish list.

Unfortunately, if the software assurance process is working right, the main benefit is that *absolutely nothing happens* [Anderson 2001, Kitchenham 1996]. And in a world of razor-thin margins, a set of activities that drive up corporate cost without any directly identifiable return is a tough sell, no matter how seemingly practicable the principle might be [Anderson 2001, Ozment 2006, Park 2006].

The business case for software assurance is therefore contingent on finding a suitable method for valuation—one that allows managers to understand the implications of an indirect benefit such as assurance and then make intelligent decisions about the most feasible level of resources to commit [Anderson 2001, McGibbon 1999].

When submitting any type of business case to your manager or to your organization's investment board, there must be a cost/benefit analysis. But it also helps to be able to answer the simple question of "Why now?"

Why now?

The world is moving forward at an amazing pace with increasing dependence on information and communication technology (ICT), yet it is still very much a nascent industry. Nations are taking the security of their national infrastructures very seriously and along with industry are making significant investments in cyber security, as well as incurring costs in responding to security breaches. To many advocates of software assurance, this investment is justified by concerns about the cost of failure.

This situation is not sustainable. As this cost of failure continues to rise, the expectation of the market will change, demanding better software and better software assurance. The government may intervene and demand higher levels of assurance in public sector procurement or increase regulation.

Your business case for software assurance may be clear, simply from the results of a cost/benefit analysis. Where it is not clear, it is important to understand the consequences of doing nothing. Software assurance is not a quick fix problem, and the longer the inevitable is postponed, the harder and more costly the solution is likely to be.

1.3 How to Use This Guide

In reading through this guide, it will become obvious that there is no single best method to make the business case for software assurance. This guide contains a variety of mechanisms, and each organization using the guide must decide on the best mechanisms to use to support strategies that are appropriate for their situation. In Section 2 we present a number of different models for computing cost/benefit. In Section 3 we discuss measurement and the need for measurement to support cost/benefit and ROI arguments. Section 4 discusses risk. Section 5 discusses prioritization, once the risks are understood. Section 6 discusses process improvement and its relationship to software assurance and business case. Section 7 discusses the topic of offshoring and its relationship to software assurance and business case. Section 8 discusses organizational development in support of software assurance and business case. Section 9 provides case studies in support of business case, and Section 10 provides our conclusions and final recommendations.

2 Cost/Benefit Models Overview

In order to calculate the costs and benefits associated with improved secure software engineering techniques, appropriate models are needed to support the computation. Here we describe a number of cost/benefit models. This discussion is largely derived from the article on cost/benefit models on the Build Security In website [Bailey 2008a]. In addition to the discussion of models here, there is a discussion of cost/benefit calculations in Appendix B.

2.1 Traditional Cost/Benefit Models

Several general models for assessing the value of an IT investment already exist [Cavusoglu 2006, Mahmood 2004, Brynjolfsson 2003, Mayor 2002]. It is our belief that the factors underlying these models can be used to build a business case for deciding how much investment can be justified for any given assurance situation [Cavusoglu 2006].

In this section we summarize the concepts and principles promoted in these models and provide a brief discussion of their common features. Below, we present the 13 most commonly cited models for IT valuation. We gleaned this list through an exhaustive review of the published ideas concerning IT valuation. Although this set is generally comprehensive, it does not encompass every approach, since the details of several models are not publicly available. However, based on our review, we believe that generic models for valuation can be factored into four categories:

- Investment-Oriented Models
- Cost-Oriented Models
- Environmental/Contextual-Oriented Models
- Quantitative Estimation Models

2.2 Investment-Oriented Models

2.2.1 Total Value of Opportunity (TVO) – Gartner

TVO is a standard metrics-based approach invented by Gartner. Its aim is to judge the potential performance of a given IT investment over time. It centers on assessing risks and then quantifying the flexibility that a given option provides for dealing with each risk. (Gartner defines flexibility as the ability to create business value out of a particular option.) TVO is built around the four factors described below [Apfel 2003]:

- cost/benefit analysis
- future uncertainty
- organization diagnostics
- best practice in measurement

Cost/benefit analysis - Total cost of ownership (TCO) is always used to characterize the overall cost of operation. Benefits are then judged using a broad range of organizational performance measures. The recommended mechanism for benefits analysis is Gartner’s Business Performance Framework [Apfel 2003]. The cost/benefit analysis must be comprehensive and appropriate to the situation, and it must describe the business case in terms that a non-IT executive can understand [Apfel 2003].

Future uncertainty - Because IT investment rarely produces immediate benefits, TVO also requires the business to quantify any probable future impacts of a given investment [Apfel 2003]. This aspect is particularly attractive in the case of software assurance, because much of the investment in securing software is designed to ensure future advantage by preventing undesirable events. These benefits should be quantified based on assumptions that can be validated retrospectively or on data-based prospective estimates such as trend line analysis [Mahmood 2004].

Organization diagnostics - These are the heart of the TVO approach. Any alteration in practice implies some form of substantive change, and organizational diagnostics essentially test an organization's ability to adapt to that change. The three types of risks associated with change—business, management, and technology—are assessed on five factors [Apfel 2003]: Strategic Alignment, Risk, Direct Payback, Architecture and Business Process Impact. Those factors coincidentally happen to be Gartner's Five Pillars of Dynamic Benefits Realization.

Best practice in measurement - This factor simply requires the employment of a commonly accepted methodology to obtain the value estimates that underlie the Future Uncertainty factor [Apfel 2003]. The aim of the measurement process is to enable a conventional business analysis that is capable of communicating the value proposition to a general audience. The key to this part of the approach is a small set of agreed-upon business metrics. The use of common metrics ensures understanding between major stakeholders. Consequently, the development of those metrics is critical to the process.

2.2.2 Total Economic Impact (TEI) – Forrester

Like TVO, TEI is meant to integrate risk and flexibility into a model that will support intelligent decisions about IT investment. TEI is a proprietary methodology of the Giga Group that allows an organization to factor intangible benefits into the equation by assessing three key areas of organizational functioning [Wang 2006]:

- flexibility
- cost
- benefits

Flexibility - Flexibility is a function of the value of the options the investment might provide. It can be described in terms of enhanced financial value or increased communication potential or on the basis of potential future increases in business value [Wang 2006]. TEI quantifies these factors using another more explicit methodology, such as Real Options Valuation (ROV) (described later). The supporting methodology can describe the actual value of the options that are available at the decision point, or it can describe the value of an option to be exercised later (for instance, an assumption that the future market share will increase as a result of an increase in assurance).

Cost - The cost analysis takes a TCO-like approach in that it considers ongoing operating costs along with any initial capital outlay. It factors both IT budget expenditures and the allocated cost of the overall organization control structure into the assessment. (The latter enforces IT accountability.)

Benefits - Benefits are expressed strictly in terms of increased business value. That expression includes any value that can be identified within the IT function as well as any value that is generated outside of IT. Thus, benefit assessments also look at the project's business value and strategic contribution and consider how appropriately the investment aligns with business unit goals.

Once these factors are quantified, the organization seeks to determine the risks associated with each of them [Wang 2006]. The risk assessment is expressed as an uncertainty or likelihood estimate that includes the potential economic impact of all major assumptions. In essence, the decision maker must be able to express both the consequences of all assumptions as well as their probability of occurrence in quantitative terms. A statement of the level of confidence in the accuracy of the overall estimate should also be provided [Wang 2006].

TEI is one of the softer kinds of value estimation methodologies and seems to be most useful when an organization's aim is to align a technology investment with a business goal or to communicate the overall value proposition of an initiative. TEI's primary purpose is to underwrite sound business decisions, given a set of alternatives [Mayor 2002]. It does that by communicating each alternative's full value in business terms. Thus, TEI can be used to justify and relate a proposed direction to any other possible directions. That creates a portfolio view of the entire IT function, which enables good strategic management practice. Since understanding the overall impacts is obviously one of the primary goals of any software assurance valuation process, TEI is an attractive approach.

2.2.3 Rapid Economic Justification (REJ) – Microsoft

In order for it to be acceptable, the cost of the software assurance process has to be justifiable in hard economic terms. But more important, that estimated cost/benefit must be available when needed. The problem is that most valuation techniques require long periods of data collection in order to produce valid results [Microsoft 2005].

The aim of Microsoft's REJ is to provide a quick and pragmatic look at the value of the investment, without taking the usual lengthy period of time to collect all the necessary operational cost/benefit data [Microsoft 2005]. Like the Total Economic Impact approach, REJ seeks to flesh out traditional TCO perspectives by aligning IT expenditures with business priorities [Microsoft 2005].

REJ focuses on balancing the economic performance of an IT investment against the resources and capital required to establish and operate it. The focus of that inquiry is on justifying business improvement [Konary 2005]. Thus, REJ involves tailoring a business assessment roadmap that identifies a project's key stakeholders, critical success factors, and key performance indicators [Konary 2005]. The latter category comprises only those indicators needed to characterize business value. The REJ process follows these five steps [Microsoft 2005, Konary 2005]:

Step One: Understand the Business Value. The aim of this step is to create an explicit map of the proposition so that both IT and business participants have a common perspective on the implications of each potential investment. That activity is proprietary to the REJ process and involves the use of a Business Assessment Roadmap that itemizes

- key stakeholders
- their critical success factors (CSFs)

- the strategy to achieve business goals
- the key performance indicators (KPIs) that will be used to judge success

Step Two: Understand the Solution. In this step, the analyst works with the owners of key business processes to define ways of applying the technology to ensure a precise alignment with the organization's CSFs. This analysis is always done in great detail, since the aim is to specify an exact solution.

As with the other models, the benefit calculation goes well beyond TCO. The analyst uses the business's commonly accepted practices to characterize process flows [Konary 2005]. The cost of each process is described from the initial planning outlay, to implementation and maintenance costs, to long-term operating expenses. The aim is to describe the investment in terms of its overall life-cycle cost and then profile that cost against all the potential benefits that might be accrued during that time [Konary 2005]. Then, REJ provides an exact quantification of the solution's value in hard financial terms [Microsoft 2005].

Step Three: Understand the Improvements. The unique feature of REJ is that it allows the organization to look beyond the traditional areas that IT might influence in order to ascertain that all potential business tasks, functions, and processes that might be improved by the prospective investment have been identified and characterized. This analysis must cross over all the functional areas and consider the potential benefits to both the IT function and those functions outside of IT, such as inventory, sales, and marketing [Microsoft 2005, Konary 2005].

Step Four: Understand the Risks. This step requires an accurate profile of all the potential risks, including their likelihood and impact. The key for this step is to factor the risk mitigation solution into the benefit and cost estimates [Konary 2005]. Doing so lets the organization optimize the economic impact of the step they are planning to take. A variant on this is to factor cost into a risk-based model and use the risk model to prioritize software assurance strategies [Feather 2001].

Step Five: Understand the Financial Metrics. Finally, all aspects of the proposed investment are characterized on a conventional financial basis, such as Net Present Value. REJ aims at building a bridge between IT and business executives [Microsoft 2005]. Thus, the terminology used to communicate the business value must ensure that all stakeholders (business and IT) can be committed to both the process and the results [Konary 2005].

2.3 Cost-Oriented Models

2.3.1 Economic Value Added (EVA) – Stern Stewart & Co

EVA approaches IT investment as a value proposition rather than as a cost. That is, EVA attempts to describe all the ways a prospective investment might leverage organizational effectiveness [McClure 2003]. EVA approaches this question by looking at a function in terms of the cost savings it might create when compared to the cost of obtaining the same function through external providers at a market rate (e.g., the cost if the service were provided by an outside vendor) [McClure 2003, Mayor 2002]. Once the comparative market value is determined, EVA quantifies the difference between the market price and the actual cost of providing the prospective function. That difference is the net operating benefit [Pettit 2001].

Costs are characterized by such things as capital outlay and opportunity cost (i.e., the potential cost of *not* doing something else). The aim of an EVA comparison is to determine whether the market value of any investment, after the actual costs are deducted, is positive [Pettit 2001]. Therefore, EVA requires a careful accounting of all expenditures as well as an honest estimate of any opportunity cost [McClure 2003].

An EVA analysis demands that everything from initial cash outlays to maintenance and training—including any expenditure that is legitimately part of the initiative—is charged against profit. EVA is then calculated as the Net Operating Profit After Tax (NOPAT) minus the Weighted Average Cost of Capital (C) as adjusted by a range of proprietary adjustments (K) that are provided as a service by Stern & Stewart [McClure 2003].

Those adjustments include such things as the "amortization of goodwill or capitalization of brand advertising." The advantage of EVA is that it produces a single financial index that can be used to characterize a diverse set of potentially contradictory directions [McClure 2003, Pettit 2001]. Approached as a tradeoff between total investment cost and potential value, EVA is a good way to gauge the impact of any process such as assurance on overall profitability. Beyond the general cost/benefit view however, EVA is really only useful when it leads into the use of another more precise valuation methodology [Mayor 2002].

2.3.2 Economic Value Sourced (EVS) – Cawly & the Meta Group

EVS sets out to quantify the value gained for every dollar invested [Meta Group 2000]. The investment in software assurance is always speculative because the risk and reward structure is hard to quantify. For instance, how do you assign a quantitative value to the increased customer trust that a secure software assurance function provides [Meta Group 2000]? In response to questions like that, EVS extends the analysis beyond the EVA approach by factoring risk and time considerations into the equation [Mayor 2002].

EVS assumes that IT investment decisions can be valued based on three strategic factors: reduction of risk, increase in productivity, and decrease in cycle time [Meta Group 2000]. Traditional return on investment (ROI) measures such as risk reduction savings or marginal productivity increases are the typical basis for quantifying value.

In addition, EVS adds standard timing factors such as flexibility. For instance, EVS asks such questions as "If the investment represents continuing cost, how quickly can those costs be adjusted to decreases in profitability?" [Meta Group 2000]. Finally, risk-based considerations, such as the overall impact of the proposed investment on performance, interoperability, resiliency, or security of the operation, are also factored in [Meta Group 2000].

EVS is an attractive approach because it allows for considerations outside of the traditional economic rate of return—considerations through which many of the indirect, abstract, or qualitative economic benefits of investment in software assurance can be understood and justified.

2.3.3 Total Cost of Ownership (TCO) – Gartner

Total Cost of Ownership (TCO) is one of the older, and more traditional, cost-based valuation approaches. It assesses an investment based strictly on its total direct and indirect costs. TCO aligns those costs with ongoing business performance in order to evaluate total value but does not assess risk or provide a means to ensure alignment with business goals [Mayor 2002].

When incorporated with a classic financial analysis such as ROI, TCO can provide a true economic value for any given investment. TCO takes a holistic view of total organizational cost over time. Ideally, it will let the manager calculate a projected rate of return on any investment based on the initial capital outlay, as well as all the aspects of the continuing cost of operation and maintenance [West 2004]. That cost estimate typically includes such ancillary considerations as physical space, security and disaster preparedness, training, and ongoing support. That's why TCO is sometimes referred to as Total Cost of Operation [Bailey 2003].

Benefit is generally calculated using an estimate of the cost that would accrue if a function or service were absent. For instance, TCO asks what the cost to the organization would be if a system failed or experienced a security incident. It then treats that cost as a risk avoidance benefit [West 2004]. By treating incident cost that way, TCO provides a good running benchmark of the financial value of an overall risk mitigation program for software assurance.

TCO can be used to monitor the overall effectiveness of any assurance program by comparing the running cost of maintaining a given level of security to existing financial data about the cost of the incidents the program is designed to prevent [Mayor 2002]. For instance, if a given level of assurance is established to prevent buffer-overflow attacks, the national average cost of those attacks can be used as an index of the benefit that would be gained by preventing them.

Because it is strictly cost centered, TCO is best used for cost rather than value estimation. However, TCO also works well in conjunction with methodologies such as the Balanced Scorecard to provide an easy to understand picture of the cost side of the proposition.

2.4 Environmental/Contextual Models

These methods, sometimes called heuristic models, add subjective and qualitative elements to the mix. Their aim is to assign a quantitative value to such intangible qualities as environmental or contextual influences, including factors such as human relations considerations and the affects of other organizational processes.

2.4.1 Balanced Scorecard – Norton and Kaplan

The Balanced Scorecard, conceived by Robert Kaplan and David Norton [Kaplan 1993], is arguably one of the easiest and most popular valuation approaches. Kaplan and Norton wanted to integrate traditional financial indicators with operational metrics and then place the results within a broader framework that could account for intangibles such as corporate innovation, employee satisfaction, and the effectiveness of applications [Kaplan 1996].

At its core, the Scorecard seeks to establish a direct link between business strategy and overall business performance [Berkman 2002]. It does that by balancing the standard financial indicators against essential, but more fluid, qualitative indicators such as customer relationship, operational excellence, and the organization's ability to learn and improve [Berkman 2002]. Thus, the Balanced Scorecard allows for ongoing assessment of the value of intangibles [Berkman 2002]. Furthermore, by requiring that every operational step be traceable to a stated strategic goal, it facilitates decisions about changes to that resource as conditions change [Kaplan 1992].

In practice, the organization's "scorecard" is customized for each operation by means of a planning process whose mission is to develop measures that capture primarily nonfinancial perspectives. Since this customization depends on the situation, there is no fixed set of quantitative measures. However, in every case, there are three or four appropriate metrics for each of the four scorecard perspectives, which are (1) financial, (2) customer, (3) internal business process, and (4) learning and growth. These perspectives are described in more detail on the Management and Accounting website [Martin ND].

The important point about using the Balanced Scorecard is that its metrics do not come in a "one size fits all" form. Generally, they come in three types. The first type includes those used to describe internal technical functions. Such a description is needed to judge technical performance against strategic goals. Examples of this type of metric include highly focused items such as reliability, processing speed, and defect rate [Mayor 2002]. These measures are not particularly useful to nontechnical managers, but they are objective and easy to aggregate into information that can help technical managers assign value to the IT function [Berkman 2002].

The second type of metric comprises those that normally come in the form of comparisons or "report cards" and are intended for use by senior executives [Kaplan 1992]. For example, if software assurance is considered a cost center, the goal is either to show how those costs have improved over time or to describe how they compare with similar costs in similar companies [Kaplan 1992]. Examples of concrete measures in this area include personnel or service costs broken out on a per-user or other kind of index basis [Berkman 2002].

The final type of metric includes those intended for use by the business side of the company [Berkman 2002]—things such as demand and use statistics, utilization analyses, and cost and budget projections. These measures almost invariably tend to be unique to each business unit [Kaplan 1992].

The important point, however, is that the Balanced Scorecard allows an organization to value all of its assets appropriately. This is essential if the organization wants to prioritize and assign security protection to the full range of those assets, not just the tangible ones. With that goal in mind, an organization can begin to collect data or analyze existing information formulated from discrete measures to support the relative valuation of its information assets.

2.4.2 Customer Index: Andersen Consulting

Andersen Consulting's Customer Index method is aimed at helping companies determine the true economic value of any particular investment by referencing it to the customer base. It does that by tracking revenue, cost, and profit on a per-customer basis. The Customer Index collects data about those items and actively associates that data with changes on a per-customer basis [Eisenberg 2003].

The organization can use this index to estimate how a prospective decision might influence the various elements of its customer base. That estimation helps the organization determine the overall value of any investment by indexing it to how it has affected, or will affect, its customer base [Eisenberg 2003]. That requires the company to calculate the current cost and profitability of all of its functions on a per-customer basis. The index allows the company to estimate what any prospective investment might do to those numbers [Eisenberg 2003].

This approach isn’t typically relevant to companies with just a few customers, but it is appropriate for any company where customer satisfaction drives every aspect of the business. More importantly, it has the potential to rationalize software assurance in terms that are intuitively realistic to business executives, whose primary goal is to increase market share [Mayor 2002].

Thus, the ability to differentiate the value of a certain set of assurance practices for a given product in terms of the impact on the customer base is a very persuasive argument for any business case. Nevertheless, the additional cost of maintaining a continuous and accurate accounting of revenue and expense on a per-customer basis is a serious consideration in adopting this approach.

2.4.3 Information Economics (IE) – The Beta Group

IE has a strategic focus. Its goal is to force managers to agree on and rank their spending priorities at the corporate level. IE does that by forcing managers to draw specific conclusions about the strategic business value of individual initiatives [Benson 1992].

IE requires a discrete value estimate for every project [Parker 1989]. That estimate is then compared across several projects based on standard economic descriptions like Net Present Value. The benefit of IE is that it provides a total relative value for each project in the portfolio. It helps decision makers to objectively assess the value of their profile of systems side by side, which should then let them allocate resources where they can do the most good [Benson 1992, Parker 1989].

IE is based around the characterization of a hierarchy of places where benefit can be derived [Benson 1992]. At the highest level, there are intangible things such as risk reduction and enhanced ROI. Further down the hierarchy, there are also hard measures such as cost and revenue. Managers prepare a list of decision factors [Parker 1989] that clearly express the benefit as a value; for example, “reduces cycle time by ‘X’ percent [Benson 1992]. Vague statements such as “will save time” are not allowed.

These decision factors, which are often scenario driven, are evaluated individually based on their relative value or risk to the business. Intangibles such as competitive responsiveness or the value of management information are assessed against a range of contingencies [Benson 1992]. Risk is typically expressed by means of a likelihood-versus-impact analysis. In effect, strategic decisions can then be referenced to that quantitative ranking [Parker 1989].

2.4.4 IT Scorecard – Bitterman, IT Performance Management Group

This is a performance measurement system similar to the Balanced Scorecard. Its aim is to let the organization track the IT operation’s financial contribution and alignment with corporate strategies. Its overall goal is to understand the IT function’s organizational strengths and weaknesses [Leahy 2002].

This approach is different from the Balanced Scorecard in that it focuses strictly on IT. Its aim is to provide a strategic basis for evaluating the IT function that is independent of all other business or organizational considerations [Leahy 2002]. The approach is therefore bottom up from the internal IT view. The organization must clearly demonstrate how much value each IT function or process contributes to the overall business value. But effective IT financial metrics are hard to find, since IT involves so many abstract and dynamic elements. That lack of measurement is one of the main reasons why IT has traditionally been viewed as a cost rather than as a resource [Leahy 2002]. Thus, the IT Scorecard focuses its measurement activity on metrics that characterize what IT brings to the business.

The intent of this approach is to communicate the value of IT rather than its cost [Bitterman 2006]. The measures used concentrate on capturing all the leading indicators of value that support the achievement of the company's strategies; for example, how fast a help desk responds to a problem and how often that problem is fixed [Bitterman 2006].

Like the Balanced Scorecard, the IT Scorecard also introduces the concept of external comparative measures and benchmarks in order to create meaningful IT performance metrics [Bitterman 2006]. The aim of the IT Scorecard is to determine how effectively current IT resources are supporting the organization and, at the same time, to assess ways that IT can better respond to future needs.

The IT Scorecard revolves around five perspectives: mission, customers, internal processes, technology, and people/organization [Bitterman 2006]. The first step in the value assignment process is to precisely characterize what the business wants out of the IT function as well as what IT can feasibly bring to the business. That description is used to establish organization-wide consensus on the metrics that will be required to capture that value.

The metrics themselves must accommodate the fact that a change in one area can have an effect on the value of another area. Thus, most successful scorecards developed through this approach are the result of numerous iterations that work toward getting this tradeoff right [Leahy 2002]. An initial set of metrics can be evolved out of this process into a group of more sophisticated measures that give greater insight into business value. However, effective measurement programs can only be customized to the strategies they support. That is the one serious weakness in this approach. The IT Scorecard can never be used right out of the box, since it requires an organization to develop and then maintain a custom set of metrics [Mayor 2002].

2.5 Quantitative Estimation Models

2.5.1 Real Options Valuation (ROV)

Real Options Valuation (ROV) aims to put a quantitative value on operational flexibility. It allows an organization to value any investment that will underwrite or create a more relevant and responsive operation [Luehrman 1998a]. Thus, ROV can be used to value technological investment.

ROV centers on ensuring maximum flexibility in the deployment of technological assets. Using this approach, an organization can determine the value of an investment by focusing on the likely consequences of a particular action over time (assuming that these consequences can be described in probabilistic terms) [Luehrman 1998a].

In most instances, those outcomes are characterized by assumptions about future performance. However, no set of assumptions is going to provide a perfect forecast. The best approach to the ROV process is to derive a value for every feasible option [Luehrman 1998a].

As a consequence, much of ROV involves identifying every factor that might be involved in or impacted by a given decision and then estimating the likelihood of occurrence. Thus, ROV is based on

1. decision variables - assumptions that are under the specific control of the decision makers and can be adjusted to increase project value as required
2. stochastic assumptions - assumptions that are random variables with known or estimated probability distributions
3. deterministic assumptions - assumptions that are based on established benchmarks [Luehrman 1998b]

Real options have concrete outcomes. Thus the decision rules for a exercising a real option must be referenced to observable behaviors that can be used to assess the performance of every variable associated with it. These behaviors must be observable and documented for a given period prior to the point at which the decision is made [Luehrman 1998a]. For example, a decision to add an assurance practice might be based on the known occurrences and costs of the threats that practice was meant to address over the past year of operation [Neely 2001].

The problem with ROV is that it is, by necessity, complex, so it works best in situations that are well defined or where experience exists. Thus, ROV models are effective in estimating the likelihood of stock options or pork bellies [Luehrman 1998b]. However, since the process of assurance is not yet well understood, the construction of the finite model for it is, at best, an exploratory effort [Neely 2001].

2.5.2 Applied Information Economics (AIE) – Hubbard

AIE is perhaps the most rigorously quantitative methodology in this set [Kwon 2001]. It centers on the use of probabilistic models to reduce uncertainty [Hubbard 1997]. It is assumed that if the appropriate amount of data can be collected (or estimated), it is possible to calculate the fiscal value of any option [Hubbard 1997].

Since all decisions involving deployment of the software assurance function involve the estimation of probabilities of both benefit and failure, it is hypothetically possible to build a sufficiently accurate picture of the financial risks and returns of any given decision option, or a related set of options, using AIE. This will allow the decision maker to understand the exact probabilities of success. This knowledge can then theoretically allow decision makers to balance their assets and activities in such a way that they will exhibit the best risk-reward characteristics [Hubbard 1997].

The analysis process itself involves classic actuarial estimation. Actuarial statistics are used in order to quantify the consequences of a given decision, which provides a proper understanding of risk and return.

Applied Information Economics computes the value of additional information. The aim is to model uncertainty quantitatively and then compute the value of marginal uncertainty reductions [Hubbard 1999]. The AIE process is based on Hubbard's Clarify, Measure, Optimize approach [Hubbard 1997], which aims to isolate and clarify the precise set of variables that are involved in and affect the decision. Such isolation and clarification allows AIE to provide specific information for decision makers.

For example, most decisions about software assurance are made based on the probability of harm. Thus, a manager might estimate that a given program would have a likelihood of 20% of failing or being exploited. AIE would restate that estimate in terms of the probabilities that a certain type of virus would be able to exploit that code, versus the likelihood that it could be compromised by a range of other attack types [Hubbard 1997]. This sort of detail makes it easier to estimate the long-term value of the decision to increase or decrease the assurance activity.

AIE analysis is considered by its proponents to be the only truly scientific and theoretically based methodology available. Its ideal outcome is an actuarial risk-versus-return statement about the probabilities of the success of a given decision [Mayor 2002]. In order to do that, AIE integrates classic principles of economics, actuarial science, and decision theory into a single approach that theoretically supports proper decision making about how to conduct business operations.

2.5.3 COCOMO II and Security Extensions – Center for Software Engineering

COCOMO II, a cost estimation technique that dates back to 1991, is the flagship for software engineering economics. It consists of a hierarchy of three increasingly detailed and accurate forms. It was designed by Barry Boehm to give an estimate of the number of programmer-months it would take to develop a software product.

COCOMO has been revised extensively over the past 25 years, and security extensions are still being developed for it. Those changes and extensions, which are risk-characterizing factors, are plugged into the model to obtain the estimates. The security components are delimited by the 13 security functions defined in ISO 15408, which is generally called the Common Criteria [Colbert 2002]. These security functions produce a standard Evaluated Assurance Level (EAL) that can be compared across products. Nevertheless, the intent of the security extensions is to simply use those criteria categories as the basis for defining the expected functionality, rather than produce an EAL [Colbert 2002].

The estimation itself is driven by a set of stock adjustment factors in the same fashion as the classic COCOMO process. Essentially, software size and security size are factored into an estimate of the total amount of LOC programmer hours (or cost) required to produce it. As with traditional COCOMO, a properly calibrated process will provide an explicit estimate of the cost that will be required to add a given amount of software functionality to the project [Madachy 2002].

There are several problems with the COCOMO approach. First, it has little recognition outside of the software engineering community, so it has to be “popularized” with traditional managers. Second, because the multiplier factors should be calibrated to the environment, COCOMO does not work in unstructured operations. Thus, it is essential that the operations they are applied to execute in a systematic and reliable way. Since the term “chaos” seems to best fit the situation in most commercial software operations, the second problem is a showstopper.

Finally and most importantly, COCOMO is too explicit to be useful as a general process cost estimate. As it is now constituted, COCOMO provides an estimate of the effort cost of adding additional security functionality to a piece of software. It does not embody variables that factor in the additional cost of the software assurance process per se. If those costs were to be added, they would obviously be part of the multiplier factors themselves. However, since the proper set of activities to secure software is presently not known or agreed on, the effectiveness of the

COCOMO II security estimation process is still awaiting proof. There is some indication that the risk factors themselves are useful in identifying areas of potential exploitation [Madachy 2002]. However, the ability to actually valuate those factors is not yet advanced enough to be reliable.

2.6 Some Common Features

Although these models represent a range of approaches, they share some common elements that should be noted for the purpose of value estimation.

2.6.1 General Factors

Holistic representation - First and perhaps most important, almost all of these models incorporate qualitative, business-oriented considerations along with quantitative factors such as cost. These considerations and factors are expressed primarily as risk versus return, but the consideration of non-tangible items, such as business priorities, is also built into the process.

Quantitative risk assessment - Risk assessment in a probabilistic sense seems to be a critical driver for almost all of these models. In that respect, the value of the investment is expressed in terms of the degree of risk avoidance that a software assurance activity can demonstrate.

Continuous execution - Valuation and risk assessment is a continuous process in every one of these models. That is because the threat environment is constantly changing, and thus, the assurance requirement is dynamic. The risk assessment is meant to support the efficient deployment of resources to mitigate priority threats to any asset of value. It should therefore be systematic and rigorous and must be an institutional process within the organization’s business model.

Standard metrics - the importance of a standard set of metrics, mutually agreed on and commonly understood, is a common thread in all of these models. Therefore, any valuation process has to begin with the development of a standard set of measures that are consistently applied across time in the practical valuation process. These measures must be maintained appropriately over time and updated immediately as conditions change.

2.6.2 Common Factors Across Models

Flexibility - Any process that enhances an organization’s ability to recognize and respond appropriately to events as they arise appears to be valuable. This flexibility is characterized by the detailed understanding of all relevant decision options and the existence of enough information about each one to support making the right choice.

Likelihood - The ability to accurately estimate the likelihood of occurrence is essential. That implies the need for enough focused baseline operating data to support stochastic estimates. It also implies the requirement to develop commonly agreed on metrics prior to the actual statistical forecasting process and to collect sufficient standard data to support estimates of probability for any valuation activity.

Granularity - When it comes to the level of focus, there are two opposing trends in these models. First, there is a trend toward value estimation based on high-level alignment with business goals and prioritization of requirements. Second, there is a trend toward decomposition of the decision process into its constituent variables at the lowest practical level of understanding. The first trend supports quicker understanding but lacks precise valuation. The second trend supports more accurate valuation but requires intensive data collection.

Decision criteria - All decision rules must be stated explicitly. Since valuation primarily involves subjective assessment, the role of the decision criteria is to provide a common basis for understanding the implications of any given proposition. Criteria are soft in the sense that they have to be developed, so it is essential that all criteria are documented prior to any operational valuation activity.

Business value - Intangible value has to be quantified. This can be done through a number of subjective methods including Delphi, business owner benchmarking, or anecdotal observation with averaging. Regardless of the approach used, the subjective value estimate has to be systematically executed and rigorously controlled. However, the principle benefit of software assurance is expected to be increased business value, which must be measured in some objective sense.

Table 1 compares the models according to these factors.

Table 1: Comparison of Cost/Benefit Models

Valuation Approach	Standard metrics	Flexibility	Type of Risk Estimate	Granularity	Decision criteria	Business value concern
Total Value of Opportunity (TVO)	cost risk	decision factor	high need	high level	explicit	central
Total Economic Impact (TEI)	other models	decision factor	uncertainty based	high level	confidence based	very central
Rapid Economic Justification (REJ)	cost/risk	not an issue	business risk	high level	cost based	central
Economic Value Added (EVA)	economic	not an issue	market based	moderate level	investment based	very central
Economic Value Sourced (EVS)	investment	affect on production	productivity based	high level	tradeoffs	central
Total Cost of Ownership (TCO)	cost/benefit	not an issue	cost/benefit	low level	investment based	investment centered
Balanced Scorecard	strategic factors	important factor	not explicit	very high level	tradeoff factors	strategic decision making
Customer Index	customer data	not an issue	customer impact	very high level	ranking impacts	customer satisfaction
Information Economics (IE)	priorities	factor	not explicit	very high level	ranking priorities	strategic consensus
IT Scorecard – Bitterman	strategic IT	important factor	not explicit	very high level	tradeoff factors	strategic decision making
Real Options Valuation	variables/ date	not an issue	not explicit	very low level	quantitative	value estimate calculation
Applied Information Economics (AIE)	probabilistic	certainty/ uncertainty	quantitative	very low level	quantitative	certainty calculation
COCOMO II and Security Extensions	COCOMO measures	not an issue	not explicit	moderate level	LOC estimate	programming cost

2.7 Limitations

Note that many of these models assume that we can accurately predict the probability of an event. As we all know, predicting the probability of a cyber attack can be difficult. Oftentimes, the best we can do is to produce rough estimates on the basis of previous data. Since attackers do not want to be detected, previous data on attacks is often incomplete, and hence the associated predictions can be based on flawed data.

Calculating risk is not as straightforward as some of the models suggest. To do it, you must have an understanding of the actual threats, vulnerabilities, and probability of exposure. Such data is not easy to come by. Moreover, the nature of the risks will change as we shift from individual hackers trying to get attention to criminals motivated by financial gain or terrorists with other motivations.

It is important to note that some experts, such as Bruce Schneier, believe that it is not feasible to accurately calculate the benefit that is derived from improved security. He points out that there is very little actual data on the cost of a break-in and that predicting the cost of a rare but damaging event is fraught with peril [Schneier 2008b].

2.8 Other Approaches

Many software security efforts are not driven by traditional cost/ benefit calculations but by reputation and customer expectation. A good example of this is the Microsoft Security Push, discussed in our case studies in Section 9. Benefits are measured in terms of reduced levels of patches and favorable customer feedback. Presumably, reduced patching saves money and favorable customer feedback aids reputation as well as keeping the customers on board.

Other organizations use risk analysis to determine which software development projects will receive more attention to software security. These organizations tend to be concerned with competitive advantage, financial loss, and reputation.

3 Measurement

"To measure is to know"

and

"If you can not measure it, you can not improve it"

Lord Kelvin

As so eloquently stated by Lord Kelvin, measurement is a foundational aspect of management, and this holds true in software assurance activities as well. The role of measurement in making a business case for software assurance comes at many levels. Measurement is key to success at the technical, operational, and strategic levels. Measurement for measurement's sake, however, is wasteful. The key success factor is the connection of measures and metrics to the business process at the correct level to make a decision. Appropriate sets of metrics for each of the levels—technical (tactical), operational, and strategic—can be defined and used to improve management of the processes leading to software assurance.

Software assurance is a subject that has different meanings to different organizations. There are many sources of definitions of software assurance, including CNSS, DoD, DHS, NASA, and NIST. Regardless of the specifics of any particular definition, they all have a common element: software assurance provides some form of justifiable confidence that the software is correct as built. To obtain justifiable confidence, all levels of an organization must be involved and execute in a properly managed manner. Following Lord Kelvin's argument, this means that each level must have appropriate measures to enable management of the software development process.

There are numerous levels through which one can view software assurance and associated metrics. Metrics can be applied at the technical level of the actual code, at an operational level associated with the code or the organization, and at the organizational level of the firm as a whole. Software assurance can also be viewed from three perspectives: a development perspective, a customer perspective, and a market perspective. Each of these perspectives has some unique characteristics that influence the use of metrics and management of software assurance. Software development is a process that is driven by management's actions to deliver required capabilities on time and within budget. The development perspective covers all of the activities associated with the actual creation of software. One example would be counting bugs, or defects per thousand lines of code. These metrics are directly connected to code quality as built. Whether a bug becomes a security defect is a larger issue, but in general a higher number of defects will result in a higher number of vulnerabilities.

3.1 Characteristics of Metrics

Metrics are a representation of something that is measurable in a process. The basis for a metric is some measurable construct that is used as a measurement. A metric is then some representation of the measurable quantity, either by itself or in combination with some other measurable quantity. For instance the metric ROI is a ratio of the gain or loss from an investment relative to the investment. The measures are the gain or loss and the size of the investment. ROI is a specific combination of these measures.

For a measurement and resulting metrics to be useful, some specific characteristics are necessary:

- quantifiable, repeatable, meaningful, robust (not subject to gaming or subversion)
- aligned with measurement objective
- useful over time periods for comparison and trending

Measurements need to be quantifiable and repeatable if they are to have any accuracy in use. Natural variation and measurement error are both normal and unavoidable. For a measurement to be useful, these disruptive elements must be at a level low enough to permit detection of meaningful change in the measurement.

For a measurement to be useful with respect to management of a process, the additional characteristics of meaningfulness and robustness are needed. *Meaningful* relates to the measurement's ability to actually track some aspect in a causal manner, as opposed to coincidence. For the measure to be safe to use, it must be robust against gaming or subversion on the part of the process users.

When a measure or a metric is to be used in the management of a process, there must be some form of alignment between the measurement and the objective of the process. If I am interested in error rates in programming, then counting the number of programmers may be measurable, but it not necessarily associated with the desired objective. Additionally, metrics guide management decisions over time, and hence they need stable time periods for reasons of comparison and trending. Changing how one measures a process over time results in a series of measurements that do not allow comparisons of performance over time.

3.2 Types of Metrics

For metrics to be aligned with the business objectives, there must be different types of metrics aligned with different levels of business management. Business can be broken into three basic levels: the strategic decision level of senior management, the operational decision level associated with mid-level management, and the tactical decision level associated with the actual work process. To match these levels, metrics can be divided into three groups:

- Organizational metrics – metrics that are related directly to the attainment of business objectives in a global strategic sense
- Operational metrics – metrics that directly support optimization of multiple work processes
- Technical metrics – metrics that directly relate to technical security details and support a specific work process

Examining each of these from the bottom up, technical metrics are those closest to the actual development environment. Software errors or bugs can be counted at numerous places in the development process, and these counts over time provide direct feedback to the development team as to their coding skill. As noted in the previous section, errors found after the software has been distributed to end users are significantly more expensive than those found during the development process itself.

3.3 Specific Measurements

Ask someone how to measure information security and you will frequently get the response "CIA"—confidentiality, integrity, and availability. The answer really does not match with the definition of metrics and measurement, for there is no explanation of how to measure compliance with respect to these concepts. The same problem exists with respect to software assurance—the objective may be provable compliance with correctness of function, but this does not explain how to measure it. Examining this from a quality perspective, we could use the concept of zero defects, which translates as no vulnerabilities, and this leads to a measurable quantity.

Measurements and metrics can serve two specific purposes with respect to management of a process. They can provide point in time data, and they can provide trend data. Point in time data refers to the use of a single measurement or metric to determine current compliance with a specified target objective. A measurement can be made and based on its value a specific decision can be made; i.e., pass or fail a test can determine whether a module moves to the next process step. Trend data can provide information associated with the state of the process over time. Data consistently collected over time allows management to make decisions associated with processes as they exist with respect to other conditions. For instance, defect rates with respect to a specific type of defect can be measured and trended, and then the efficacy of a specific training event to reduce that defect can be determined by examination of trends before and after training.

Examining the cost/benefit models from the previous section yields some common business measures found in many business processes and ones that might be reasonably expected in managing software assurance. Two examples will be compared, one based on a quality measure and the other on ROI. For each of these examples, the specific measurements at each level will be examined to illustrate how they can be derived and supported.

A key element in any manufacturing process is the control of defects. Zero defects may be a catchy mantra, but it is not typically practical in an economic sense. Defects with respect to coding of software can be counted in many ways; bugs discovered during testing, missing implementation of requirements, coding not to specified coding standards, and compiler errors and/or warnings. Because not all "bugs" are equally important, it is common practice to also assign a severity to each specific item according to some ordinal scale: critical, high, moderate, or low. This classification allows resources to be directed toward the more important issues and can lead to rules such as "All critical and high-level bugs will be addressed before release to production." To implement a bug count, several tactical process elements are required. There needs to be some formalized mechanism for determining that a bug exists and another for determining the relative severity of the bug. The first aspect, bug determination, can occur at several points in the development process: at unit test, subsystem test, and full system testing. This determination can also occur after production from field reports. For reasons of continuity and consistency, severity determination should use the same criteria regardless of bug determination.

The determination of a specific level of defect, say a critical defect, can act as a point in time measurement that changes the production process, i.e., prevents the code from proceeding to the next level of the development process until correction. The documentation and trending of the same data over time allows management to see trends and determine longer term actions. Bugs may be tracked by originator, as in specific developer or designer, and a series of repeated bugs may result in a determination of a need for retraining of a specific worker or team. Bugs can also be tracked by type, using a form of enumeration such as the CWE [MITRE 2009], resulting in the ability to determine whether specific types of bugs tend to be repeated.

The measurement of bugs as in bug counts can be converted to a metric such as bugs per thousand lines of code (kloc) or bugs per time period since code release, to enable comparison across products and releases. At the tactical, developer level, the measurement of bugs makes workable sense, as it provides direct feedback that can be used for management of the immediate development process. Using the metric form of bugs/kloc allows higher levels of management to manage resources across projects and teams to minimize defect rates. To assist in this type of effort, trend charts illustrating this data by project and team or time period allows management to examine the effectiveness of specific decisions. Should management notice a rise in a specific type of defect over time, or of defects in total by team over time, specific targeted actions can be taken to attempt to remedy the problem. The effectiveness of the actions can also be determined by examining data trends before and after the action.

To determine a metric such as ROI, the measures needed are costs/savings and investments. These are financial measurements that can be calculated based on the collection and attribution of activity based cost data to the software development process. To calculate the ROI of testing, one could calculate the cost of fixing bugs without development testing and compare to the cost of fixing bugs with development testing and determine the return on the investment in testing. This case is fairly obvious, as data has consistently shown that defects that are corrected after shipping are significantly more expensive than those corrected during production, in some cases by orders of magnitude. This same concept can be applied to many events across the development process; the ROI of a specific tool can be calculated based on the cost change associated with the deployment of the tool and resulting change in defect correction costs. Although common in many industries and well accepted in financial circles, ROI suffers from one underlying weakness. Its accuracy depends upon the precision and accuracy of attribution of specific cost data. To get detailed information, very fine grained detail into activity based costing is required. Taking a developer’s time sheet and coding it into categories such as coding, testing, defect correction (by defect type), etc. is not a simple and accurate process. Inaccuracies in the underlying data and the ability to game the system through the allocation and assignment of cost data has resulted in questionable accuracy of ROI in certain types of environments, including software development. Add in the complexity of not knowing a defect may exist until later discovery and the whole usefulness of this measure frequently gets called into question.

Secure software development is a process with significant levels of documented success stories [Howard 2006, Microsoft 2009]. As a process, there are many measures and metrics that can be made associated with the process itself, and not just the outcome of the process. The measurement of number of people trained and the associated metric percentage of people trained can be used to manage desired states of personnel awareness or training. The number of modules and percentage of modules passing or failing specific process points can be tracked and managed. The list of measurable and hence manageable items can be large. Further examples can be found in documents such as the *Practical Measurement Framework for Software Assurance and Information Security* report from the DHS Software Assurance Measurement Working Group [Bartol 2008].

Software assurance has wider ramifications than just those associated with the development process. Firms may be concerned about software assurance associated with procured software. This shifts the concern and management from a manufacturing mode to an acquisition mode and brings in issues such as managing contracting with respect to cost and other factors. Deciding contract issues based on terms such as cost, schedule, and performance to requirements is common. If one of the performance requirements is the design to a specified level of software assurance, or based on some software assurance process, acquisition personnel will require training in the topic to enable them to ensure the aspect is used in the acquisition decision. This makes process-based measurements and metrics such as number and percentage of contracting personnel trained with respect to software assurance standards meaningful from a management perspective. Outcome-based measurements such as number and percentage of programs procured that comply with management's desired software assurance level can be used to track direct compliance with desired objectives in this area.

3.4 What to Measure

The ability to count things and make measurements in a process is never the challenging element in a metrics driven management process. The challenge comes from determining the correct items to measure to support desired management objectives. This places the mantle of responsibility upon upper management to determine the appropriate and desired level of security associated with software assurance. This declaration then enables middle level management to determine appropriate measures and metrics to support their management of processes to achieve the desired objectives. The more mature an organization becomes with respect to understanding and communicating desired software assurance objectives, the more effective metrics and measures can be in assisting management in obtaining those goals.

A recent trend has been to reduce the attack surface area of a software product. This can be done through the application of threat modeling techniques and then redesign and recoding to reduce the number of entry and exit points of code. Once middle management embarks on this course, a whole series of specific measurements and metrics can be deployed to assist in this effort. Counting the number of entry and exit points and charting versions of software can give an indication of changes in this aspect. Tracking the use of threat modeling by project and/or module is another metric that can aid in attaining a reduction in threat exposure.

A robust software assurance program will have multiple avenues of managing the complexities associated with developing and using software in a secure manner. For each separate aspect of management, there needs to be some measure or metric to assist in the determination of effectiveness. This leads to a hierarchy of metrics associated with software assurance, with different yet specific metrics at each level of management. At the bottom of the process, the individual worker, there may be some specific measures associated with that person or team's impact to the desired level of software assurance. At each level of management, some form of aggregation is associated with the metric to apply it to the next level if it is still associated with an objective at the new level. Training is a prime example. Each employee has training requirements and responsibilities associated with their specific role in the organization. Each employee can calculate the percentage of required training that they have achieved with respect to software assurance. As you rise through the organization by level, training level achievement may be aggregated not only by units of employees, but also by topics of training. At the top of the organization there could be a measure percentage of training attainment. The objective is the alignment of measurement to management objective at each level.

3.5 SDL Example

One of the greatest values of metrics is the ability to compare results of processes before and after changes to make a determination of the effectiveness of an improvement effort. To measure the effectiveness of the development process, bug count is one potential measurement. To refine this measure and make it useful, it can be refined as follows. Bugs come in many varieties: critical, serious, major, minor, and inconsequential. This classification is useful, as resources may not permit all bugs to be addressed, so it is important to focus on the more damaging. Bugs also take time to discover, so the overall quality of the development process is more accurately determined after a period of time, such as a year. Microsoft used this specific measure to demonstrate the value of its SDL process, showing that critical and serious bugs discovered in the year after release were dramatically reduced after the adoption of the SDL. Using Windows 2000 (pre-SDL) as a baseline and Windows 2003 (post-SDL), critical and serious bugs reported in the year after release were 62 and 24, respectively, a > 60% decrease [Lipner 2005].

Using the same measures for SQL Server 2000, pre- and post-SDL, has values of 16 and 3. Exchange Server 2000 pre- and post-SDL had values of 8 and 2 respectively [Lipner 2005]. Again, the measures show dramatic improvements, and the measures also illustrate another issue on comparative measures: you must compare like-to-like to have meaningful comparisons. Because of the complexity of the various code bases and the differences in requirements and design, it is not meaningful to compare the actual values between products.

This example demonstrates the value of metrics and measurement in the software assurance arena. Software development processes are complex, multistep events, with numerous teams involved and many variables, some controllable, some not. Process improvement efforts are important to improve the business, but their effectiveness can’t be proven without measurement. Gross before and after measures do not necessarily invoke specific causality, yet as shown in the example, they can provide useful data to the organization.

4 Risk

4.1 Introduction

Business leaders, including those responsible for deciding where investment dollars are spent, use risk management as a significant decision making structure for identifying and mitigating all categories of enterprise and organizational risk. In recent years, risk management has been successfully applied in determining where to invest IT and information security dollars. Due to the increasing marketplace demand to develop and field more secure software, risk management processes, methods, and tools are being tailored and applied to software and application security. Such use is intended as a mitigation to help reduce the costs associated with security incidents caused by insecure software, including ongoing patching costs.

Risk management is particularly useful when the demand for investment resources substantially outweighs the supply, calling for a structure that defines meaningful business thresholds and ensures that organizational exposure does not exceed them. Thus risk management and risk assessment can be effectively used to drive the business case for building security in to newly developed software and determining where and when to upgrade operational and legacy software.

Those with the authority to fund software assurance activities need to be able to answer the question Why should I (my organization, my project) spend this money? and subsequently Where and how should I spend this money? The cost of software assurance is the *additional* cost incurred to guarantee that the software is free from exploitation and harm by any adversary [Bailey 2008b]. Decision makers must understand risk well enough to decide whether the benefit of additional investments in software assurance practices sufficiently outweighs the cost when compared to other risk mitigation actions that are already funded and in place. Once this risk cost/benefit decision has been made in favor of additional software assurance, managers must determine where and how they should spend the money. Software assurance practices can be applied enterprise-wide or to a specific development project; this tradeoff is also part of the investment decision making process. Risk is the principal and most commonly used indicator to determine which software assurance practices should be rolled out to which parts of the organization. In this sense, risk informs the decision making process.

The increasing integration of business processes and IT systems means that software risks can often be linked to serious and specific impacts on the mission of an organization or business. Since 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 clearly 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 spur action. The risks we focus on in this report are all tied directly to software and all have clear security ramifications. However, unless these risks are described in terms that business people and decision makers understand, they will not likely be addressed.

All businesses understand the principles and practices of risk management at some level. Executives make use of risk management concepts on a daily basis. Strategic decisions and tactical moves are often informed with risk management data. Business dashboards include business risks as critical measures. Yet software development has not traditionally leveraged this understanding of risk management to gain a clear business mandate.

4.2 Risk Definitions

Many definitions of risk exist. NIST Special Publication 800-30 *Risk Management Guide for Information Technology Systems* [Stoneburner 2002] defines risk as

> *a function of the* ***likelihood*** *of a given* ***threat-source's*** *exercising a particular potential* ***vulnerability****, and the resulting* ***impact*** *of that adverse event on the organization.*

NASA-STD-8719.13B[1] defines risk as

> *the combination of (1) the probability (qualitative or quantitative) that a program or project will experience an undesired event and (2) the consequences, impact, or severity of the undesired event were it to occur*

ISO/IEC 27005 *Information technology – security techniques – Information security risk management* [ISO 2008a] defines information security risk as

> *potential that a given threat will exploit vulnerabilities of an asset or group of assets and thereby cause harm to the organization*

Alberts [2002] defines risk as “the possibility of suffering harm or loss.” Jones [Jones 2005b] defines risk as “the probable frequency and probable magnitude of future loss.”

While all definitions emphasize likelihood or probability, NASA-STD-8719.13B calls out the fact that there are qualitative and quantitative ways of determining probability. NIST SP 800-30 and ISO 27005 emphasize the presence of a vulnerability as a source of risk. These sources attempt to define *what* risk is. This is a necessary precursor to figuring out *how* to identify risk, analyze risk, and mitigate risk as a significant basis for making the business case for software assurance.

4.3 A Framework for Software Risk Management

Management of risks, including the notion of risk aversion and technical tradeoff, is deeply impacted by business motivation. As a result, software risk management can only be successfully carried out in a business context. Software security risk includes 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.

A necessary part of any approach to ensuring adequate software security is the definition and use of a continuous risk management process. An effective approach is using a well-defined risk management framework (RMF) to assess and mitigate risk throughout the software development life cycle. The RMF described in this section can be used to implement a high-level, consistent, iterative risk analysis that is deeply integrated throughout the SDLC [McGraw 2005].

[1] http://www.hq.nasa.gov/office/codeq/doctree/871913B.pdf

Figure 1 shows the RMF as a closed loop process with five activity stages:

- Understand the business context.
- Identify the business and technical risks.
- Synthesize and prioritize the risks, producing a ranked set.
- Define the risk mitigation strategy.
- Carry out required fixes and validate that they are correct.

Throughout the application of the RMF, measurement and reporting activities occur. These activities focus on tracking, displaying, and understanding progress regarding software risk.

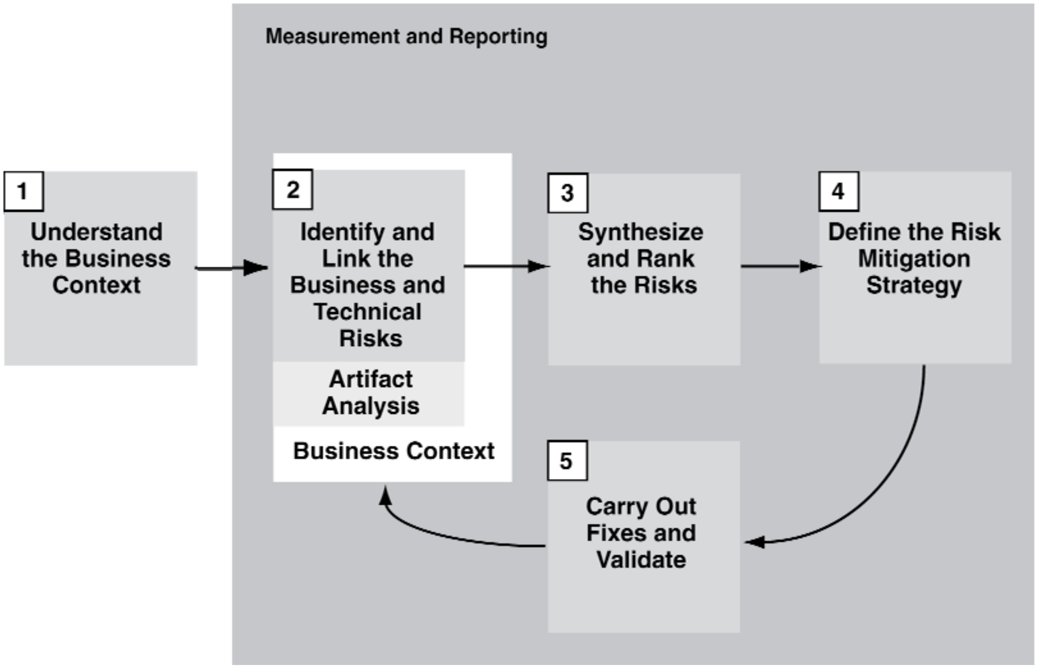

Measurement and Reporting
1
Understand the Business Context
2
Identify and Link the Business and Technical Risks
Artifact Analysis
Business Context
3
Synthesize and Rank the Risks
4
Define the Risk Mitigation Strategy
5
Carry Out Fixes and Validate

Figure 1: A Software Security Risk Management Framework

4.3.1 Understand the Business Context

The first stage of the RMF involves getting a handle on the business situation as described in Section 4.1, Introduction. Commonly, business goals are neither obvious nor explicitly stated. In some cases, a business may even have difficulty expressing these goals clearly and consistently. When applying the RMF, the analyst must extract and describe business goals, priorities, and circumstances in order to understand what kinds of software risks to care about and which business goals are paramount. Business goals include, but are not limited to, preserving and enhancing reputation, increasing revenue, meeting service level agreements, reducing development costs, and generating high return on investment.

A key part of understanding the business context is determining the criteria that the business has established for accepting risks and acceptable levels of risk, often referred to as risk tolerances or risk thresholds. “Risk thresholds are a management tool to determine when risk is in control or has exceeded acceptable organizational limits. For example, a risk threshold for virus intrusions may be whenever more than 200 users are affected—this would indicate that management needs to act to prevent operational disruption” [REF 2008]. During this stage it is also critical to define the scope of the risk assessment and ensure that there is a clear and direct tie to business and mission objectives.

4.3.2 Identify the 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 impact 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 increase in development costs. The severity of a business risk should be expressed in terms of financial or project management metrics. These may include, for example, market share (percentage), direct cost, level of productivity, and cost of rework.

Business risk identification helps to define and guide the use of particular technical methods for identifying, analyzing, and mitigating software risk for various software artifacts such as requirements, architecture, and design specifications. The identification of business risks provides a necessary foundation that allows software risk (especially impact) 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, a technical 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. The process an organization follows may offer opportunities for mistakes in design or implementation. Technical risks involve impacts 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.

Refer to Section 4.4 for further details on candidate risk assessment methods and Section 4.5 for more detailed guidance on identifying risks.

4.3.3 Synthesize and Rank (Analyze and Prioritize) Risks

Large numbers of risks 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 analyzing and prioritizing risks, the critical Who cares? question can (and must) be answered. Analysis 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 impact the business. This stage creates as its output a list of all the risks and their relative priorities for resolution. Typical risk metrics include, for example, risk likelihood, risk impact, risk severity, and number of risks emerging and mitigated over time.

Refer to Section 4.6 for more information on analyzing risks and Section 4.7 for detailed guidance on categorizing and prioritizing risks.

4.3.4 Define the Risk Mitigation Strategy

Given a set of prioritized risks from stage three, the next stage 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 directly 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, return 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 risks, so some level of residual risk will remain once mitigation actions are taken. Residual risks need to be regularly reviewed and consciously managed.

Refer to Section 4.8 for further guidance on mitigating risks.

4.3.5 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 four. 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 validate that software artifacts and processes no longer bear 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.

4.3.6 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. Measurements against this master list should be 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.

Links to descriptions or measurements of the corresponding business risks mitigated can be used to clearly demonstrate the business value of the software risk mitigation process and the risk management framework. Such measurements can aid managers to

- 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

For a detailed example of applying the RMF to a software server product, refer to *Software Security: Building Security In*, Chapter 2 [McGraw 2006]. For another comprehensive description of a software-based risk management framework, refer to Microsoft's *Security Risk Management Guide*.[2]

The following sections provide details that elaborate the stages of a continuous software risk management process as described by the RMF. These include (1) methods for assessing risk, (2) identifying risk, (3) analyzing risk, (4) prioritizing risk, and (5) mitigating risk.

4.4 Methods for Assessing Risk

Methods for finding, analyzing, and mitigating risk abound. Some methods are considered qualitative; others are considered quantitative. Some methods focus on risks to specific assets (such as software); others focus on groups of assets (such as IT infrastructures). Some methods rely on the subjective judgment of domain experts; others rely on surveys or structured interviews involving participants from across the organization.

Vorster and Les Labuschagne [Vorster 2005] developed a framework for evaluating different risk methods. They evaluate two qualitative methods:

- OCTAVE (Operationally Critical Threat, Asset and Vulnerability Evaluation) [Alberts 2002]
- CORAS (Construct a platform for Risk Analysis of Security Critical Systems) [Stolen 2002]

And three quantitative methods:

- ISRAM (Information Security Risk Analysis Method) [Karabacak 2005]
- CORA (Cost-Of-Risk Analysis)[3]
- Information Systems (IS) analysis based on a business model [Suh 2003]

They present scales of criteria that can be used to decide which method best fits an organization.

Other relevant methods include

- CRAMM[4] (CCTA Risk Analysis and Management Method), which uses interviews, objective questionnaires, and guidelines and provides a range of assessment tools for modeling asset dependencies and performing business impact analysis (CCTA is the Central Computing and Telecommunications Agency of the United Kingdom government.)
- ERAM[5] (Enterprise Risk Assessment Methodology) (U.S. Department of Defense), which uses a collaborative review process with sponsors, program office personnel, and acquisition experts

[2] http://technet.microsoft.com/en-us/library/cc163143.aspx

[3] The references cited for CORA in [Vorster 2005] are no longer available but the method description remains useful.

[4] http://en.wikipedia.org/wiki/CRAMM

[5] http://www.bta.mil/products/eram.html

- GRAM[6] (Gartner Risk Assessment Methodology), which uses a Delphic technique to obtain expert opinions, rather than interviews or workshop
- NIST *Risk Management Guide for Information Technology Systems* [Stoneburner 2002]
- STRIDE[7] (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege), which is based on using a set of defined attack patterns to elicit risks that may be present in an existing software design
- Threat Modeling [Howard 2006, Chapter 9; Goertzel 2007, Section 5.2.3.1], which uses structured interviews, inspections, and expert analysis

Most organizations engaged in risk management use a combination or a customized version of these methods. That said, it is difficult to assess the success of various approaches due to the absence of publicly available experience reports or case studies. In their 2007 Broad Area Announcement for Cyber Security Research and Development,[8] the U.S. Department of Homeland Security stated the following:

> *The lack of sound and practical security metrics is severely hampering progress both in research and engineering of secure systems. DHS seeks metrics that offer an understanding of the costs and benefits of alternative approaches to securing systems. Ideally, DHS wants to be able to choose research and engineering courses of action that have the highest risk reduction return on investment, i.e., that reduces risk most for the lowest cost.*
>
> *DHS specifically does not seek metrics which have no validatable connection to the ultimate goal: the reduction of risk to mission.*

Regardless of the method chosen, it should address the following categories of risk:

1. Operational risks that arise from [REF 2008]
   - poorly designed, poorly executed, and failed internal processes
   - inadvertent or deliberate actions of people, including accidental disclosures, modifications of information and software, insider threat, and fraud
   - failure of systems to perform as intended or risks imposed by the complexity and unpredictability of interconnected systems
   - failure of technology, including the unanticipated results of executing software or the failure of hardware components
   - external events include failures in mergers, acquisitions, and supply chain and other third-party relationships, particularly with parties providing software. External events also include forces such as natural disasters and failures of public infrastructure.
2. Legal, regulatory, and compliance risks that arise from failures to comply with the laws and regulations of international, national, state, and local governments. One such example is the U.S. Department of the Treasury Office of Comptroller of the Currency (OCC) Bulletin 2008-16[9] that directs specific action by financial institutions for ensuring application secu-

[6] http://www.gartner.com/it/page.jsp?id=782612

[7] http://msdn.microsoft.com/en-us/library/ft0y04t6(vs.71).aspx; (also see [Howard 2003])

[8] https://www.fbo.gov/index?s=opportunity&mode=form&tab=core&id=70da95bedbf9a8b44ffae4436d0daefa&tab=documents&tabmode=list

[9] http://www.occ.treas.gov/ftp/bulletin/2008-16.html

rity. Compliance risks can also arise from failure to comply with mandated standards and compliance frameworks such as the ISO 27000[10] series, COBIT[11] (Control Objectives for Information Technology), and PA-DSS[12] (Payment Application Data Security Standard). There are a growing number of regulations calling for stringent protection of personally identifiable information (PII). Additional legal risks can arise from failure to comply with valid contracts as well as civil and criminal investigations and e-discovery requests.

3. Reputational risks that arise from public disclosure or incidents that affect the public's perception of an organization's soundness, credibility, and reliability. A PII exposure, the mishandling of a security incident, or an impaired ability to provide expected products and services can all have a significant negative impact on an organization's reputation. The realization of reputational risks can result in loss of market share, declining stock prices, and loss of customer confidence.
4. Financial risks arise from having to incur costs to address risk in all of the categories above. Expending funds to recover from a realized risk is a lost opportunity, given that such funds are not available to address more proactive mitigation actions.

The sections following provide a general overview of how risks can be identified, analyzed, and mitigated. No particular method is preferred over another, but threat modeling is used to illustrate these risk practices. This discussion will likely need to be translated or tailored when considering each unique method, given the differences in the focus and emphasis of each.

4.5 Identifying Risks

Risk identification is a foundational risk management activity. It requires the organization to identify and assess the types and extent of threats, vulnerabilities, and disruptive events that can pose risk to the viability of high-value assets such as software. Identified risks form a baseline from which a continuous risk management process such as RMF can be established and managed [REF 2008].

The identification of risks involves several steps:

- An organization starts by identifying what software assets are important. The intent is to focus the risk assessment on software assets judged to be most critical or of the highest value for mission success. Software asset owners are also identified; these are typically software project managers or software line of business/product managers. Software asset interrelationships and interdependencies are identified so they can be considered during the risk assessment.
- The organization next determines what might threaten high-value software assets. They then determine candidate vulnerabilities (both organizational and technological) and how these could be exploited by identified threats, causing risks to be realized.
- The organization finally determines the impacts due to losses resulting from realized risks.

[10] http://www.iso27001security.com/html/iso27000.html

[11] http://www.isaca.org/cobit

[12] https://www.pcisecuritystandards.org/security_standards/pci_pa_dss.shtml

There are many techniques that can be used to identify risk. These include [REF 2008]

- use of questionnaires and surveys
- interviewing key management personnel and subject matter experts
- review of process controls
- conducting software security risk assessment using one or more of the methods described in Section 4.4
- performing internal audits and performance reviews
- performing business impact analysis
- performing scenario planning and analysis
- using risk taxonomies for similar organizations and industries
- using lessons-learned databases such as the security incident knowledgebase
- reviewing vulnerability catalogs such as those maintained by CERT[13] and CVE[14]

Questions designed to elicit risks "should address *business risks* (such as motivation, market, resources, schedule, people, facilities, budget, contracts, program interfaces), *project risks* (such as development process, development system, management methods, work environment) and *product risks* (such as technical defects, design flaws, bugs, issues with languages and platforms). Particular effort should be made to address questions regarding risk indicators, the likelihood that risks may occur, and business impact estimates in case risks are realized" [McGraw 2006].

4.5.1 Assets

An asset is anything that is important to the business. This could be money, customer information, strategic and operational plans, process definitions, designs, formulas, etc. Assets can be both tangible (facilities, IT infrastructure) and intangible (intellectual property, customer data, software). The proliferation and use of software intensive systems for key business services means that secure software is increasingly important when determining which assets are high-value and thus require ongoing management and assessment of their risks.

The risk assessment methods introduced above identify assets in several ways. CORAS relies heavily on workshops with IT personnel and focuses on technical risk using UML class diagrams for each asset. Therefore, assets are defined as systems or components of systems. One type of threat modeling [Howard 2006[15]] identifies assets by looking at business objectives, roles, data, and components. Another type of threat modeling [Ingalsbe 2008] identifies assets by looking at specialized UML deployment diagrams which call out roles, data, and interconnected systems.

4.5.2 Threats

A threat is "an indication of a potential undesirable event" [NSTISSC 1998], "the actor or agent who is the source of the danger" [McGraw 2006], or "anything that is capable of acting against an asset in a manner that can result in harm" [Jones 2005b]. In other words, a threat is something that could adversely affect a software asset. Threat classification serves as a foundation for risk identi-

[13] http://www.kb.cert.org/vuls/

[14] http://cve.mitre.org

[15] Also http://msdn.microsoft.com/en-us/security/aa570413.aspx

fication and assessment. In the information assurance field, threats are classified as being capable of violating confidentiality, integrity, or availability (CIA). This classification is useful as a starting point but insufficient for software assurance because it focuses on threats to information, not threats to software. However, at least one type of threat modeling uses this approach for classifying threats [Howard 2006]. Examples of software assurance threats include "threats to code integrity, intellectual property protection, personally identifiable information (PII) or sensitive data, features that require admin/root privileges, and external network interfaces" [SAFECode 2008].

Threats can also be classified using a threat taxonomy like the Computer Security Incident Information Taxonomy [Howard 1998] or an attack pattern taxonomy like the Common Attack Pattern Enumeration and Classification (CAPEC[16]). After deciding on a classification scheme, threats to assets must be identified. Although virtually all risk methods require the identification of threats, there is very little documentation that prescribes how to identify threats. The threat classification schemes provide some help in this area. For example, Microsoft's STRIDE (see Section 4.4) is a way of classifying threats that could be used to create a list of starter questions for assets or groups of assets (for example, "Is there any way that someone could spoof someone's identity in order to gain access to this asset?") Obviously, this is insufficient. One type of threat modeling [Ingalsbe 2008] prescribes a way of identifying threats by using starter questions for roles, data, systems, and nodes. Although not optimal, involving knowledgeable software security experts during the process of threat identification is still the best way to ensure that all significant threats are identified.

4.5.3 Vulnerabilities

A vulnerability is a "defect or weakness in system security procedures, design, implementation, or internal controls that can be exercised and result in a security breach or a violation of security policy" [McGraw 2006]. Sometimes vulnerabilities are software defects (a buffer overflow in C++ code, for example). Sometimes vulnerabilities are the result of using software-intensive systems in a way that they were never intended (for example, connecting 20 year old SCADA systems (Supervisory Control And Data Acquisition) to an intranet). Vulnerabilities can be discovered by looking at each threat, identifying a specific vulnerability that would allow the threat to be realized, and then creating a threat-vulnerability pair that is then ranked and addressed [CMS 2002]. Vulnerabilities can also be discovered by creating threat trees that document the sequential set of conditions that must be true in order for a threat to be realized [Howard 2006]. Vulnerabilities can be discovered using tools that statically or dynamically inspect software and systems. Online databases and communities of practice can be used to identify vulnerabilities in known technologies such as Common Vulnerabilities and Exposures and Common Weakness Enumeration.[17] Finally, vulnerabilities can be discovered by involving experts in the vulnerability identification process.

4.5.4 Impacts to Assets

Impacts to software assets occur when a threat successfully exploits a vulnerability, causing a risk to be realized. An IT person might talk about asset impacts like this: "The denial-of-service attack caused the network to go down." A business person might talk about asset impacts like this: "The attack caused us to be unable to process credit applications for four hours." The second impact is

[16] http://capec.mitre.org/

[17] http://cve.mitre.org/; http://cwe.mitre.org

a result of the first but is stated in terms of the asset (credit applications), not the technology (network). These are two valid ways of looking at assets, but it is important to understand that impacts to assets should be discussed from a business perspective when building a case for software assurance.

4.6 Analyzing Risks

The analysis of risks involves “establishing relationships between the business goals, business risks, and technical risks and subsequently prioritizing them in meaningful business terms” [McGraw 2006]. An organization starts by determining the business impact of a risk being realized (resulting from a threat exploiting an asset vulnerability). Next they attempt to determine the likelihood of the risk being realized. Finally, they calculate a risk rating.

4.6.1 Business Impact

A defined, consistent, repeatable process must be used to determine the impact of an event. Annual Loss Expectancy (ALE[18]) and Single Loss Expectancy[19] can be used to determine damage, but dollar amounts are sometimes difficult to predict. DREAD[20] (Damage Potential, Reproducibility, Exploitability, Affected Users, Discoverability) is used by several threat modeling methods and is fairly straightforward. A qualitative high, medium, or low value can be used for damage potential (i.e., business impact). Stoneburner et al. [Stoneburner 2002] define the following levels for potential impact:

- High: Exercise of the vulnerability (1) may result in the highly costly loss of major tangible assets or resources; (2) may significantly violate, harm, or impede an organization’s mission, reputation, or interest; or (3) may result in human death or serious injury.
- Moderate: Exercise of the vulnerability (1) may result in the costly loss of tangible assets or resources; (2) may violate, harm, or impede an organization’s mission, reputation, or interest; or (3) may result in human injury.
- Low: Exercise of the vulnerability (1) may result in the loss of some tangible assets or resources or (2) may noticeably affect an organization’s mission, reputation, or interest.
- Additional factors often must be considered to determine the magnitude of impact. These may include
  - an estimation of the frequency of the threat-source’s exercise of the vulnerability over a specified time period (e.g., one year)
  - an approximate cost for each occurrence of the threat-source’s exercise of the vulnerability
  - a weighted factor based on a subjective analysis of the relative impact of a specific threat’s exercising a specific vulnerability

[18] The expected monetary loss that can be expected for an asset due to a risk over a one year period. http://www.riskythinking.com/glossary/annualized_loss_expectancy.php

[19] The expected monetary loss every time a risk occurs. http://www.riskythinking.com/glossary/single_loss_expectancy.php

[20] http://www.owasp.org/index.php/Threat_Risk_Modeling#DREAD

Determining a meaningful value for business impact that can be compared across all risks is invaluable when determining whether software assurance practices make sound business sense.

4.6.2 Likelihood

The likelihood, or probability, that a risk would be realized is often difficult to predict. Historical data and trend analysis of past events can help. A high, medium, low scoring approach is common. One such approach is described in [Stoneburner 2002]:

- High: The threat-source (actors) is highly motivated and sufficiently capable, and controls to prevent the vulnerability from being exercised are ineffective.
- Medium: The threat-source is motivated and capable, but controls are in place that may impede successful exercise of the vulnerability.
- Low: The threat-source lacks motivation or capability, or controls are in place to prevent, or at least significantly impede, the vulnerability from being exercised.

4.6.3 Risk Valuation

A valuation is assigned to each risk using either qualitative (high, medium, low) or quantitative (dollar value) factors. These factors are determined based on asset impact, business impact, likelihood, and any other meaningful factors used in the selected risk assessment method (refer to Section 4.4). Stoneburner et al. [Stoneburner 2002] present one example as follows:

> *Table 2 shows how the overall risk levels of high, medium, and low are derived. The determination of these risk levels or ratings may be subjective. The rationale for this justification can be explained in terms of the probability assigned for each threat likelihood level and a value assigned for each impact level. For example,*
>
> *The probability assigned for each threat likelihood level is 1.0 for high, 0.5 for medium, 0.1 for low.*
>
> *The value assigned for each impact level is 100 for high, 50 for medium, and 10 for low.*

Table 2: Risk-Level Matrix

Impact	**Low** **(10)**	**Medium** **(50)**	**High** **(100)**
Threat Likelihood			
High (1.0)	Low 10 X 1.0 = 10	Medium 50 X 1.0 = 50	High 100 X 1.0 = 100
Medium (0.5)	Low 10 X 0.5 = 5	Medium 50 X 0.5 = 25	Medium 100 X 0.5 = 50
Low (0.1)	Low 10 X 0.1 = 1	Low 50 X 0.1 = 5	Low 100 X 0.1 = 10

Risk Scale: High (>50 to 100); Medium (>10 to 50); Low (1 to 10)

4.7 Categorizing and Prioritizing Risks

Grouping risks into meaningful categories can help prioritize risks for disposition and mitigation. Analysts merge similar statements of risk and eliminate redundant ones. Related risks are grouped for efficient handling and the cause and effect between related risks is identified [REF 2008].

A relative priority is determined for each risk statement (or risk statement group) based on the assigned risk valuation. Priorities determine which risks need the greatest attention. Stoneburner et al. [Stoneburner 2002] describe one risk scale and commensurate actions, as shown in Table 3 and following the example presented in Section 4.6:

Table 3: Risk Scale and Necessary Actions

Risk Level	Risk Description and Necessary Actions
High	There is a strong need for corrective measures. For example, an existing system may continue to operate, but a corrective action plan must be put in place as soon as possible.
Medium	Corrective actions are needed and a plan must be developed to incorporate these actions within a reasonable period of time.
Low	The asset owner/project manager must determine whether corrective actions are required or decide to accept the risk, including managing the residual risk.

4.8 Mitigating Risks

“When the consequences of risk exceed the organization’s risk thresholds and are determined to be unacceptable, the organization must act to mitigate risk to the extent possible. An important part of risk management is to determine a mitigation strategy for each identified risk and to implement actions to carry out the strategy. Mitigation strategy development begins with assigning a risk disposition to each risk—a statement of the organization’s intention for addressing the risk” [REF 2008].

Risk dispositions can vary widely across the organization or project but typically include the following [REF 2008]:

- risk avoidance: altering some aspect of the software design or code to avoid the risk while still providing required functionality
- risk acceptance: acknowledging the risk but consciously not taking any action (in essence, accepting the potential consequences of the risk)
- risk monitoring: performing further research and deferring action on the risk until the need to address the risk is apparent
- risk transfer: assigning the risk to a competent outside party to manage and mitigate
- risk control and mitigation: taking active steps to minimize the risk

An organization starts by determining the mitigation strategy for each risk that requires action. Once strategies are defined and software assurance practices are selected, the organization puts a process in place to manage residual risk.

4.8.1 Mitigations

The risk mitigation process should answer the question How can identified software risks best be managed? Implementing a software assurance practice is one form of risk mitigation. The organization analyzes a range of mitigation strategies and software assurance practices. Candidate practices are associated with each risk along with an estimate of the effectiveness of each practice and the level of rigor with which each practice must be performed. Implementing software assurance practices must make sense economically and, in the best of all cases, they will have a clear return on investment that can be presented and demonstrated [McGraw 2006].

The cost of practice implementation needs to be compared with the cost associated with the realized risk if the practice was not implemented, to determine if the cost/benefit argument is sufficient. Therefore it is important to provide accurate, detailed estimates so that a true comparison can be made. Costs can be expressed in dollars, staff time, and operational effectiveness.

Each proposed practice should cover as many risks as possible. The organization also needs to assess the impact of laws, regulations, and organizational policy on its ability to implement specific practices, as well as how each practice will affect the software throughout the SDLC including during operations [McGraw 2006].

For the purposes of making the business case for software assurance, only mitigations associated with software assurance practices should be called out. For example, a mitigation for a regulatory risk might be to reject doing business in a specific country. This is not a software assurance activity. A mitigation for a technical risk might be to use a well-vetted, language-specific coding standard. This is a software assurance practice.

4.8.2 Residual Risk

Residual risk is the risk that remains and is accepted by the organization after mitigations have been implemented. Residual risk must be specifically accepted, deferred, or transferred. The organization must regularly measure, review, and potentially revise the disposition and mitigation status of all identified risks to ensure that business conditions do not call for changes in mitigation strategies. In addition, full risk assessments should be conducted periodically. This includes all residual risks, some of which may become risks that require active mitigation.

4.9 Summary

The purpose of this section has been to describe how commonly used approaches for managing risk in other disciplines (specifically information technology and information security) can be readily tailored and applied to software security. Risk management is a proven and useful strategy when the demand for investment resources substantially outweighs the supply. Thus risk management and risk assessment can be effectively used to drive the business case for building security in to newly developed software and determining where and when to upgrade operational and legacy software.

Methods for risk management can be qualitative, quantitative, or some combination of both. All methods need to consider various categories of risk to include operational risks; legal, regulatory, and compliance risks; risks to reputation, brand, and image; and financial risks. Risk assessment be-gins with risk identification, which involves determining what high-value or key software assets are most likely at risk, the threats to those assets, vulnerabilities that can be exploited by identified threats, and the potential impacts due to asset loss and violations of confidentiality, availability, and integrity. Risk impact and risk likelihood are established to help determine risk valuation. Risks are prioritized based on their respective valuation.

Based on risk priorities and valuation, the organization determines what risk actions to take, including risk avoidance, acceptance, monitoring, transfer, or control and mitigation. Control and mitigation includes determining what software security practices to implement during specific phases of the SDLC. Risk actions are continually reviewed as part of normal operational and strategic planning and reporting processes, as business conditions are constantly changing and are thus potentially impacting risk priorities and risk actions.

5 Prioritization

For business leaders and project managers, prioritization is the name of the game: determining where to pay attention, allocating limited resources, and managing risks. In today’s business climate, managers are constantly dealing with the demand to do more with less. The resources required to run the business, let alone to invest in new initiatives (such as software assurance), are always at a premium—time, money, staff expertise, information, and facilities, not to mention energy and attention span. All resource allocation decisions are about doing what is best for the organization (and its stakeholders) to achieve the mission and satisfy objectives. However, what is best is sometimes hard to define, hard to quantify, and even harder to defend when the demand for resources exceeds the supply.

Business leaders are becoming more aware of the need to invest in information and software assurance—to meet compliance requirements and optimize their total cost of ownership for software-intensive applications and systems. So how do managers ensure that investments in software assurance and security are subject to the same decision and prioritization criteria as other business investments? And by so doing, how are they able to justify investments that increase their confidence in their ability to increase operational effectiveness, as well as protect digital information using software that is more able to resist, tolerate, and recover from attack?

This section describes an approach for selecting security investments using business-based criteria: the Security Investment Decision Dashboard (SIDD). The approach and supporting tool define seven decision criteria categories, each supported by three or more indicators. Categories and indicators are ranked and applied to a series of investments. Individual investment scores are presented for discussion and evaluation by decision makers. This approach can be use to rationalize and prioritize any class of security investments, including software assurance practices.

The SIDD is one way of making well-informed security investment decisions using business-based criteria. This approach is based on recent CERT research. Over the past four years, CERT has developed a body of knowledge in enterprise and information security governance, including a detailed framework and implementation guide that describe a robust security governance program.[21] When faced with this framework of tasks, actions, roles and responsibilities, and outcomes, senior leaders say “This is all well and good, but I have many more pressing issues to deal with than security governance. Can you provide me with an aid to select and prioritize these and other security-related actions that I can use as an input to normal planning and capital investment processes?” The SIDD is such an aid.

5.1 Foundation and Structure

The Security Investment Decision Dashboard (SIDD) provides a means for evaluating and comparing several candidate security investments. A foundational principle of SIDD is that the priorities for candidate investments are driven by the organization's *desired outcome for any given investment*, not just security investments. This ensures that security investments are subject to the

[21] http://www.cert.org/governance

same decision criteria as other business investments. They can then be presented, reviewed, analyzed, debated, and compared using the same scales, factors, and investment-selection criteria and processes.

SIDD describes seven decision criteria *categories*, each supported by three or more decision *indicators*, totaling 33 in all. Two CERT reports [Allen 2005, Westby 2007] served as the starting point for selecting business-based criteria that could be used to evaluate candidate investments. A number of relevant business and security sources [Campbell 2006, CISWG 2005, Drugescu 2006, ISO 2007, Kleinfeld 2006] were analyzed for business-based questions and factors that could help inform security investment decisions. The collected set of questions and factors are reflected in the current set of 33 indicators. The seven categories were derived through affinity grouping of the 33 indicators.

Each category is described in the form of one or two questions to ask. Categories are presented in shaded text in Table 4 and include Cost, Criticality & Risk, Feasibility, Positive Interdependencies, Involvement, Measurability, and Time & Effort Required. The importance of each category is determined by considering the question What should *any* candidate investment do for the organization and its stakeholders? or alternatively, What is the basis or criteria for selecting *any* candidate investment?

For example, is it most important that an investment be (1) low cost, (2) critical to meet business objectives or mitigate a high degree of operational risk, or (3) feasible in terms of likelihood of success? The questions in Table 4 define each category. Priorities or rankings are then assigned to the category based on the importance of the category to the organization's investment selection process. Each category is further elaborated by three or more indicators that are listed following each category in Table 4. This is a “starter set” that can be tailored to a specific organization’s decision factors.

Table 4: SIDD Categories and Indicators

Category	**Indicators**
Cost	Estimated total cost to accomplish this investment, taking into account the potential cost savings and/or risk reduction to the organization
	Overt cost in dollars at outset to accomplish this investment
	Estimated life-cycle cost in dollars over time to sustain this investment
	Cost of NOT doing this investment, in terms of potential exposure and residual risk (high = investment is more necessary)
	Potential cost savings to organization beyond breakeven point, if quantifiable (ROI), over time (high = better)
Criticality & Risk	Degree to which this investment contributes to meeting the organization's business objectives and risk management goals
	Degree to which this investment is key or mainstream in helping the organization meet its primary objectives and critical success factors
	Degree of risk (as assessed in terms of likelihood and potential impact—high/medium/low priority) mitigated by this investment
	Degree to which this investment helps the organization protect stakeholders' (shareholders') interests
Feasibility	Likelihood of investment success
	Likelihood of success on first try
	Likelihood of success on subsequent tries (if first try fails)
	Likelihood that turnover among management and/or board of directors will negate work expended on this investment (low likelihood = better)

Category	Indicators
	Likelihood that this investment will need to be rolled back (low = better)
Positive Interdependencies	(1) Degree to which this investment integrates with or represents reasonable changes to existing organizational processes and practices, rather than requiring new ones (2) Degree to which this investment paves the way for future investments (compliance, policy, risk management, etc.)
	Degree to which other investments/tasks are dependent on this one (i.e., degree to which this investment makes it easier to accomplish additional tasks)
	Degree to which the accomplishment of this investment makes it easier to comply with current laws and regulations
	Degree to which the accomplishment of this investment makes it easier to comply with potential new laws and regulations in the future
	Degree to which existing knowledge and/or skills can be used to accomplish this investment, rather than requiring new skills/knowledge
	Degree to which this investment produces positive side effects (e.g., enhancing brand/reputation, building customer trust, benefiting supply chain partners)
Involvement	What level of involvement and buy-in are required from various parties for investment success—both within and outside of the organization
	Level of buy-in required throughout the organization (Must all employees be on board 100% for this to work? Or only a subset, such as management and certain key employees?)
	Extent to which this investment requires the active involvement of many departments across the organization
	Number of people who need to be actively involved
	Level of involvement by third parties required (partners, consultants, vendors, etc.)
	Degree of external, independent assessment/auditing (vs. in-house assessment/auditing) required
Measurability	How measurable the outcome of this investment is
	Degree to which this investment can be evaluated using existing approaches and reporting mechanisms
	Measurability of the outcome (Can it be quantified in tangible terms such as revenue, market share, or stock price?)
	If the outcome is intangible (e.g., goodwill, increased customer trust, enhanced brand), whether the benefits can be demonstrated against meaningful business success factors
Time & Effort Required	(1) Level of staff-hours required to accomplish this investment (2) How long it will take to reach break-even cost for this investment
	Board of directors time required
	Senior management time required
	Cross-organizational team/steering committee time required
	Middle and lower management time required
	Other key staff time required
	Time likely needed to achieve the required level of buy-in
	Time required to achieve first demonstrated results
	Time required to achieve full adoption and use of the investment results across all affected business units
	Time to achieve breakeven, if quantifiable

A manager may determine that there are other factors or different factors that they use in their investment decision making processes. The SIDD is designed so that categories and indicators can be changed, added, and deleted, and the dashboard will continue to present meaningful comparisons.

Dashboard results are presented in a comparative bar graph form (see Appendix C.3 for an example). Score totals are presented for the 7 categories and the 33 indicators for each investment. An additional result is calculated for the 6 indicators ranked highest (1-6). This result has been included to accommodate the situation where a subset of indicators is important for investment selection as a companion to the total scores for all categories and for all indicators.

5.2 Using the Dashboard

Investment priorities and comparative scores are determined using a two-phased approach. In Phase 1, a decision maker prioritizes categories (Step 1) and indicators (Step 2). The idea here is to determine the importance of each category and each indicator when making *any* organizational investment decision. These priorities (or rankings) will be applied to all candidate investments during Phase 2.

Phase 2 defines the candidate investments that are to be evaluated (Step 3). There is no upper bound but typically 3-5 investments are evaluated in one use of the dashboard. The decision maker then answers the category and indicator questions (Step 4) for each investment. Scores are calculated by applying the priorities specified in Phase 1 to these answers. Each step is further described below. An example dashboard entry for each step is presented in Appendix C.

Phase 1: Establish priorities for all types of organizational investments

Step 1

> Rank categories 1-7 (shaded entries in Table 4) based on their relative importance for any organizational investment decision, 1 being most important and 7 being least important. Do not consider any specific security investment when performing this ranking. An example category ranking appears in Appendix C.1.

Step 2

> Rank indicators 1-33 (more detailed entries in Table 4), again, based on their relative importance for any organizational investment decision, with a ranking of "1" as the most important. Given that 33 indicators is a long list to prioritize, some dashboard users grouped these into three sets of 10 and then ranked the group of 10. Others created larger scale granularity by assigning a value of, say, 1, 5, or 10 to all 33, which then produced a larger numeric difference between investment scores. An example indicator ranking appears in Appendix C.1.
>
> The current version of the dashboard does not enforce a correlation between category and indicator rankings. This means that one category could be ranked as having the highest priority, while indicators in other categories could be ranked as being more important. This correlation will be enforced in the next version.
>
> Steps 1 and 2 are intended to be done once and then applied during all subsequent investment analyses. This helps ensure that results are based on the same ranking and thus can be meaningfully compared. Rankings are periodically reviewed during normal planning cycles or following key events (such as a merger or acquisition) to ensure that they continue to reflect current business priorities.

Some dashboard users have suggested that one or more senior C-level leaders perform the category ranking and another group, such as a cross-organizational steering committee, performs the indicator rankings. When category and indicator rankings are done independently, these can then be compared to see if they are consistent or reveal misunderstandings or differences of opinion. In several cases, the shared understanding that resulted from doing these rankings was of equal or greater value than the dashboard results.

Phase 2: Evaluate each investment

For each candidate security investment:

Step 3

Define the investment so that those evaluating it have a common understanding of its scope and intent. While SIDD has been used to evaluate security governance and IT investments (such as policy development, specifying segregation of duties, developing an asset inventory, deploying wireless, creating a new operations center), four examples of software assurance investments are selected here and further illustrated in Appendix C.

- A: Integrate architectural risk analysis into the standard SDLC.
- B: Integrate secure coding practices for C and C++ into the standard SDLC.
- C: Integrate the use of static code analysis tools into the standard SDLC.
- D: Integrate security requirements engineering using Carnegie Mellon's SQUARE method into the SDLC [Mead 2008].

Deciding to use SIDD assumes that resources are insufficient to start up all of these now, so managers use this approach to help inform which ones to fund.

Step 4

Answer the category and indicator questions (Table 4) for each investment by using a dashboard worksheet, one per investment. Determining an answer for each question is accomplished by selecting a value from 1 to 5. Based on the question, answers range from very high to very low or very low to very high. An example of answers for one investment and one category is shown in Appendix C.2.

Step 5

Review and discuss the results. An example of summary results for the four example investments appears in Appendix C.3.

Dashboard outcomes identify the highest priority (highest scoring) investments based on the category rank, the indicator rank, and the answers to the questions for each investment. Given that the category and indicator ranks are fixed (and weighted to normalize the scores[22]), the dashboard results can be meaningfully compared and used to help select which investments to fund, as well as providing a defensible rationale for those that were not selected.

If, based on other factors, these highest scoring investment choices are not justified, this is a valuable opportunity to re-examine the category and indicators rankings and answers to determine if they do indeed reflect how the organization makes investment decisions.

[22] Category and indicator ranks are converted into weights that are used as multipliers to normalize dashboard scores. This is necessary due to a priority of "1" being highest, yet the highest total score reflects the highest priority investment.

This tool is not intended as a substitute for human judgment. It can be used to make judgments more explicit, to apply a consistent set of decision criteria to all investments which can then be communicated, and to capture trends over time.

SIDD can be tailored to replace current categories and indicators to ones that are more meaningful to an organizational business unit or project. It can also be tailored to use a reduced set of decision criteria. SIDD can be used to support dynamic, “what-if” decision making by changing category and indicator priorities and examining the results.

SIDD or a comparable approach can be effectively used to prioritize software assurance practice investments using the same criteria that are applied to other categories of business investment. This provides a defined, repeatable approach to help inform the investment decision process.

6 Process Improvement and Secure Software

6.1 Ensuring a Capable Process

Capable processes produce capable products. In that respect, experience has shown that substantive, well-defined processes will ensure a continuously improving organization [Paulk 1993]. The potential to improve performance through the disciplined execution of well-defined processes is the reason why the general concept of process improvement is so influential for software organizations.

All of the models that we are going to discuss in this section serve the same general purpose, which is to ensure the continuous improvement of the software organization. The outcome of the adoption of one of these models is a sustainable process that produces a significantly reduced number of exploitable defects. Better still, when a business case for software assurance is built around return on investment, it should be noted that these same well-defined processes have also been shown to provide anywhere from 300% to 2,300% ROI [Jones 2005, McGibbon 1999]. Consequently, the bonus from the use of a process improvement approach to improve software security is that it will also produce a much more productive and cost-efficient organization.

In the United States, some familiar examples of process improvement models include the Software Engineering Institute's Capability Maturity Model Integration (CMMI®) framework, along with the retired Capability Maturity Model® for Software (Software CMM). (Background information about software process improvement models and their history can be found in Appendix D.) In both of these, process capability is assessed based on the presence, or absence, of proxies. These proxies correspond to the essential practices that are generally considered to ensure against defects. Nonetheless, the role of every process improvement model is to define the structural relationships that are needed to ensure a consistent and reliable process. The model itself provides the overarching framework, within which the organization develops and deploys the actual practices that ensure the proper execution of each process.

In addition to supplying a template for the development of an organizational scale software process, these models also embody some sort of structured assessment or evaluation process. The role of the assessment is to document how well a target organization meets the requirements of best practice as defined by the general process model. That assessment then underwrites two of the most important factors in the conduct of global business, trust and competence. That is, the findings of a formal assessment can enhance the level of trust between a supplier and a customer as well as provide the customer with substantive proof that the supplier will be able to deliver on commitments.

The ability to meet commitments depends on the organization's ability to ensure predictable outcomes. Software organizations have a particularly difficult time ensuring predictability because the product itself is abstract functionality. Moreover, thanks to the internet, the actual development process can span continents and oceans. As a result, two partner organizations have little

® CMMI and Capability Maturity Model are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

basis for gauging each other’s performance unless they’ve done business before. Additionally, even given experience with each other’s work, if the process is chaotic there is no guarantee that the next project will turn out the same.

Consequently, a defined process based on a proven framework like CMMI can provide the basis for the development of optimally secure and correct software, while also assuring trust. There is also a very high correlation between the capable and systematic performance of classic best practices such as requirements management, measurement and analysis, and configuration management and the production of secure code. Therefore, these models, which were initially developed to ensure a quality software organization, can also be employed to assure a secure one. The assurance of secure products is a new purpose for process improvement models. Nonetheless, all of the approaches outlined in this section could potentially provide guidance for the development of a comprehensively and systematically secure organization.

6.2 Adapting the CMMI to Secure Software Assurance

The CMMI framework that is currently in use, Version 1.2, includes three models that apply to different types of organizations. CMMI for Development is appropriate for software organizations that perform product and service development processes. CMMI for Acquisition is appropriate for organizations that perform acquisition processes. Its body of knowledge contains advice for successful acquisition and supply chain management. Finally, CMMI for Services integrates bodies of knowledge that are essential for successful service delivery. The appropriate model has to be selected at the time of adoption. There are also two types of CMMI model representations, staged and continuous. One of these two CMMI approaches must be selected to comply with the stipulations of the model.

The continuous representation of the model is intended to allow the organization to select the implementation order for process areas (PAs) that best meets the organization’s business objectives. CMMI continuous has the advantage of enabling comparisons across and among organizations on a process area by process areas basis and provides an easy integration with ISO 15504. That is because the organization of the process areas and the common features for each process are derived from the ISO 15504 model, which will be discussed later.

The staged representation provides a sequence of staged improvements like the Software CMM, beginning from basic to advanced capability. It has the advantage of permitting comparisons across and among units based on these maturity levels. It should be noted, however, that CMMI does not organize the stages in the same way as either the Software CMM or ISO 15504. The five CMMI maturity levels are

1. Initial
2. Managed
3. Defined
4. Quantitatively Managed
5. Optimizing

The next sections describe the process areas in each maturity level.

6.2.1 Level 1 – Initial

Although there are five maturity levels in the CMMI, the first level is by definition without defined PAs. Therefore, the actual assessment of maturity comes at the next level (2). The Initial stage is chaotic, in the sense that there are no repeatable processes. Consequently, there is a high potential for the introduction of vulnerabilities into the product.

6.2.2 Level 2 – Managed

Level 2 installs a set of essential practices to ensure that the organization is following correct fundamental practice in software management. These are the seven level 2 PAs for CMMI:

Configuration Management establishes and maintains the integrity of the software item throughout the life cycle. Along with Process and Product Quality Assurance, this PA is perhaps the most critical aspect of proper organizational management. That is because it introduces the overall control element. It is not possible to ensure security in software if the process is not controlled, and so basic configuration management is a critical general requirement. However, in addition to the fundamentals of control, the elements of secure software management can be introduced here.

Measurement and Analysis gets quantitative management features into the process as early as possible. Because quantitative information is critical to ensuring secure practice, the addition of this PA greatly strengthens the effect of level 2 when it comes to secure software assurance.

Process and Product Quality Assurance enables managers to have complete visibility into the evolving software process and provides a more complete understanding of the level of quality of the products being built. In many ways, these practices are the foundation of the entire software security process. Therefore it is important to include all relevant practices to ensure that the software is free of vulnerabilities.

Project Planning establishes the operational basis for the project. This is communicated through a set of explicit plans. These plans serve as the necessary foundation for the management of the software project. However, proper planning and management is perhaps the most important single feature in the process of ensuring a secure product. Thus, standardized practices aimed at ensuring that the planning incorporates security into the overall product development can be inserted here.

Project Monitoring and Control establishes and maintains an “adequate” level of understanding of what is taking place within the project itself. This includes the monitoring of progress, so that a manager can take effective action when the project’s performance deviates significantly from the plans laid out in the Project Planning PA. Obviously, ensuring that the project steers its planned course is the other side of the coin in ensuring that standard security practices are maintained in the project space.

Requirements Management establishes the required consensus between the customer and the software supplier. However, the definition and proper translation of security functional requirements is a critical part of ensuring a secure product. Therefore, the addition of practices oriented toward understanding and mitigation of threat during requirements elicitation can leverage the overall security of the software product or service. Since software at a certain level is essentially nothing more than the sum of its requirements, the lack of a capable requirements management process means that the organization’s software functionality will be either illogically defined or inadequately controlled.

Supplier Agreement Management defines a mechanism that will let the organization select qualified software subcontractors and manage them effectively. It focuses the aims of the Requirements Management, Project Planning, and Project Monitoring and Control PAs onto any third-party organization involved in the development of the software (e.g., subcontractors). The general purpose is to establish managerial control over the process within another organization. Since the supply chain is fraught with peril from a security standpoint, the stipulation and enforcement of standard practices for suppliers is an obvious necessity for maintaining security in large, complex projects.

6.2.3 Level 3 – Defined

CMMI Level 3 installs a set of monitoring and control activities that ensure that the organization is reliably performing the practices that are associated with a well-defined and fully managed software operation. This level therefore contains a number of PAs to facilitate good management. These are the 11 CMMI PAs at level 3:

Decision Analysis and Resolution provides prospective and retrospective feedback on decision outcomes. In addition to improving short-term and long-term planning, this function is tied to some extent to formalized management of change. Both planning and evolution are critical elements of a secure process, and so a function that ensures that these are done correctly and with a security focus is a valuable addition to the overall goal of ensuring software security.

Integrated Project Management integrates the various software engineering and management operations of the organization into a coherent, defined set of practices. This program is aggregated from the process and related assets described in the Organizational Process Definition PA. The elements of this PA reference and adopt the needs of the business environment and technical requirements of the project to the particular situation. Therefore, proper integration of known security factors into this PA is critical.

Organizational Process Definition develops and maintains a defined collection of software process assets. That includes a formal description of all of the activities that were installed at level 2 as well as a formal inventory of the measures and tools used by the organization at later levels. This is the operational side of Organizational Process Focus, since it maintains a continuous picture of the processes and assets maintained by the organization. These processes would be the elements that would be involved in any formal security work.

Organizational Process Focus unequivocally establishes and assigns the responsibility for the actual refinement of the software process for the organization. This is a critical element in security because the organization’s processes themselves are ongoing and often larger than individual units or managers. Thus, accountability for their successful application and refinement is often ill defined if ever articulated at all. It is not possible to ensure a secure process without following a well-defined process, and it is this PA that establishes that process.

Organizational Training develops the skills and knowledge of the people in the organization such that they are able to carry out their roles and assignments as effectively and efficiently as possible. Security involves enhancing the skills and knowledge of the workforce, and so it is essential to incorporate proper training and education into the standard security program.

Product Integration establishes a means for the group responsible for coordinating the process to communicate with and coordinate the individual units that are involved in the process. Product Integration is the interdisciplinary aspect of Integrated Project Management. It extends well beyond the requirements of simple software engineering into the realm of governance, and thus it is an essential condition in ensuring the integrity and effectiveness of all security activities.

Requirements Development is a PA aimed primarily at ensuring that the full set of functional requirements is specified. This PA is different from the requirements management PA implemented in level 2 in that it is aimed specifically at ensuring an effective requirements elicitation and documentation process. Although the primary purpose of this PA is to make certain that the functional requirements set is correct, it applies to the development of security functional requirements as well. Thus, the addition of this feature as a PA strongly supports the effort to ensure that security is an integral part of the build.

Risk Management is a critical element of ensuring the overall security of the process and product. Therefore, the addition of this PA directly serves the overall goal of secure software. In essence, efforts to understand risk prior to embarking on any project have to be institutionalized as a well-defined and persistent organizational process. Then that understanding can be leveraged into continuous improvement of the process and product. That requires both disciplined practice and consistent documentation of outcomes.

Technical Solution turns the requirements set into a logically and provably correct design. The comments in the Requirements Development section above about the inclusion of security functionality apply in the case of design. However, there is an additional component of assurance with design. That is the issue of ensuring the integrity and consistency of the actual design. Since many of the logical defects in code appear at this level, it is critical, from a security standpoint, to ensure that the design is not only correct but also properly executed.

Verification institutionalizes peer review methods such as inspections as a formal organizational process. This traditional activity is planned and managed as a well-defined process. Since secure software is dependent on effective inspection practice, a PA that ensures inspection processes are properly deployed and coordinated is valuable to the overall goal of secure products.

Validation institutionalizes methods such as testing as a formal organizational process. This traditional activity is planned and managed as a well-defined process. Since secure software is dependent on effective testing practices, a PA that ensures testing processes are properly deployed and coordinated is valuable to the overall goal of secure products.

6.2.4 Level 4 – Quantitatively Managed

There are only two process areas at this level in CMMI. They are highly interdependent. The purpose of each of these PAs is to get and keep a comprehensive understanding of both the software product and the software process.

Organizational Process Performance applies a comprehensive measurement program to the software work products. It provides an in-depth, measurement-based understanding of the characteristics of the software organization’s products as well as its processes. It lets the software organization achieve its specific quality goals.

Quantitative Project Management adds a formal comprehensive measurement component to the overall management of software development, sustainment, and acquisition. It fosters explicit, quantitative control over the performance of the software project by providing the capability to monitor and evaluate the outcomes obtained from a given activity. The aim of this particular PA is to assess the specific causes of any anomaly within a defined software process. Then it seeks to correct the circumstances that caused the problem to occur. The ability to apply quantitative management practices to ensuring the overall security of the software process and product is critical and in some respects is the business payoff for all of the prior activities. Thus, it is important for every organization interested in secure software to get to this level eventually.

6.2.5 Level 5 – Optimizing

The PAs at level 5 itemize the concerns that both the organization and its projects must address in order to ensure continuous and measurable improvement.

Causal Analysis and Resolution does analysis on quantitative information in order to understand the causes of defects. It then institutes defined procedures, activities, or tasks to prevent their reoccurrence. This PA is data based in that analyses are done on a well-defined basis, on explicitly designated data. The process of Causal Analysis develops specific actions to alleviate an identified problem and then monitors those actions in order to ensure that the problem has not reoccurred. This process is so clearly an integral part of good practice for software assurance that no further discussion is required.

Organizational Innovation and Deployment is a formal process that explicitly ensures that the organization will continuously improve its software processes and technologies. Since the aim of this process is to ensure an optimum product, the usefulness to the business case should be implicit. This process is also data based in the sense that organizational data is used to identify and then plan the best course of action. At the same time, the data also provides tracking information to ensure proper deployment and sustainment of new processes and technologies, which facilitates the assurance of secure software.

6.2.6 Implementing the Process Areas

Process areas are what make each of the capability levels distinct. By definition, each process area exists at a single maturity level. However, as can be seen here there are obvious relationships and dependencies between process areas at the various levels of maturity. Therefore, the implementation of secure processes should not be seen as a rigid, lock-step activity. Organizations should develop secure practices within specific situations, without the requirement that they be implemented across the board in the entire operation. However, attention must continue to be focused on the implementation of a logical and mutually supporting process area structure.

6.2.7 Differences Between the CMMI and Software CMM Process Areas

In CMMI level 2, in addition to the six PAs of the Software CMM, there is a seventh PA, Measurement and Analysis. It has been promoted to the foundational level in order to get quantitative management features into the process as early as possible. Because quantitative information is critical to ensuring secure practice, the addition of this PA greatly strengthens the level 2 array.

The difference between the process areas of the CMMI and Software CMM is much more pronounced at level 3. CMMI features 11 of these instead of the 7 PAs contained in the Software CMM. The additional PAs bring level 3 more directly into line with the life-cycle process areas defined by IEEE 12207, and in essence they begin to harmonize CMM with that model. The additions are Decision Analysis and Resolution, Requirements Development, Risk Management, Technical Solution (design), Verification, and Validation.

The level 4 and level 5 CMMI process areas are very similar to those of the Software CMM. They differ primarily in the means used to express them. Level 4 features Organizational Process Performance and Quantitative Project Management instead of Quantitative Process Management and Software Quality Management. Level 5 has two PAs instead of three, although the intent is similar to that of the Software CMM. In CMMI, these PAs are Organizational Innovation and Deployment (which embodies the two change management PAs), and Causal Analysis and Resolution, which is similar to Defect Prevention.

6.3 The CMMI Appraisal Process

The CMMI SCAMPI[SM] appraisal methods are the generally accepted methods for evaluating the results of using CMMI models. The SCAMPI Method Definition Document (MDD) defines rules for ensuring the consistency of appraisal ratings across organizations. The achievement of a specific maturity level or the satisfaction of a process area must mean the same thing for any appraised organization.

SCAMPI includes Class A, B, and C appraisal methods, which vary in scope and purpose. SCAMPI A is the most rigorous method and the only method that can result in a rating. SCAMPI B provides options in model scope, but the characterization of practices is fixed to one scale and is performed on implemented practices. SCAMPI C provides a wide range of options, including characterization of planned approaches to process implementation according to a scale defined by the user.

More information about SCAMPI methods is available on the SEI website [SEI 2009a].

6.4 Adapting ISO 15504 to Secure Software Assurance

This project was initially called SPICE. It was voted up as a standard in January 2001 after spending three years at the technical report (TR) status. Its stipulated purpose is to consolidate all of the existing ISO process improvement schemes into a single model. ISO 15504 bridges the gap between an organization’s need to define a coherent set of best practices and the concomitant practical requirement that these practices can be implemented and assessed. It is not ISO’s intent to use ISO 15504 as a scheme for the certification/registration of the process capability of an organization in the same manner as ISO 9000 and 14000. Instead, ISO 15504 is intended to provide a framework for the assessment of software processes. This can easily be applied to security assessments. ISO 15504 can be seen as an attempt to learn from the successes and failures of that

[SM] SCAMPI is a service mark of Carnegie Mellon University.

previous generation of models, including the staged approach and ISO 9000. The goal of the ISO 15504 standard is to “encourage a culture of continuous improvement as well as to establish the proper mechanisms to support and maintain that culture” [ISO 1998b]. The explicit scope of application for the standard is defined by ISO as process assessment, process improvement, and capability determination. The intent of the standard is specifically aimed at

- Risk Assessment: The standard evaluates suppliers through software capability determination.
- Process Improvement: The standard helps organizations improve software development processes.
- Process Assessment: The standard offers a single assessment scheme instead of a variety of audit approaches.
- Management Information: The standard provides information that lets managers evaluate their software processes against business needs.

The stated purpose of ISO 15504 is to provide a structured approach for the assessment of software processes [ISO 1998a]

- by or on behalf of an organization with the objective of understanding the state of its own processes for SPI
- by or on behalf of an organization with the objective of determining the suitability of its own processes for a particular requirement or class of requirements
- by or on behalf of an organization with the objective of determining the suitability of another organization’s processes for a particular contract or class of contracts

The emphasis on risk and supplier capability makes this standard an ideal model for developing secure processes. The standard fosters process improvement by evaluating the effectiveness of practices embodied in a given software process to improve them. It supports capability determination by assessing whether a given set of processes satisfies the specified business needs of the company [ISO 1998a]. It is assumed that the software process constitutes the practical framework within which an effective and/or a secure process can be developed and assessed.

Because of this strong orientation toward process, the ISO 15504 documentation suite (and particularly the Baseline Practices Guide) is closely linked to ISO 12207. In fact, the ISO 15504 Standard was painstakingly harmonized with that earlier standard as an active part of its development process. Accordingly, the software process domains that are considered appropriate to ISO 15504 are the same as those covered by ISO 12207: (1) Acquisition, (2) Supply, (3) Development, (4) Operation, (5) Maintenance, and the range of 12207 supporting and organizational processes.

Nonetheless, in addition to defining a standalone process improvement activity, the standard is also adaptable to other models. It is intended to establish a “common framework for expressing the process capability ratings of a 15504 conformant assessment as well as provide a migration path for any existing assessment models wishing to become 15504-conformant” [ISO 1998a]. The key to the successful application of this standard is the assessment process, which can embody any number of definitions of best practice.

Thus, one tempting area for application of this standard would be in the rating of organizations based on identified secure software best practices. That is the exact purpose of the ISO/IEC 21827 standard, which will be discussed next. ISO 21287 uses 15504's capability assessment framework and approach. It is therefore difficult to distinguish between the two models. In fact, the only material difference is in the base practices specified by each model. ISO 15504 specifies quality practices, which as we have said might also be considered security practices, while the ISO 21827 standard specifies only those practices that are meant to ensure a secure software process.

The assessment framework of 15504 encourages self-assessment. Assessments evaluate the adequacy of the management of a target process, taking into account the context in which the assessed process operates. The outcome is a set of process ratings (a process profile) rather than the pass/fail result produced by the CMMI staged representation. This is appropriate across all application domains and organizational sizes. ISO 15504 specifically supports the Acquisition (5.1) process by

- reducing uncertainties in the selection of suppliers through identification of risks associated with contractor capability prior to contract award
- enabling appropriate controls to be put in place for risk containment
- establishing a quantified basis for balancing business needs, requirements, and estimated project cost against the capability of competing suppliers

Consequently, the benefit for acquirers lies in the ability to determine the current and potential capability of a supplier's software processes, which equates to risk in the security universe. In the case of suppliers, the benefit lies in the capacity to determine the current and potential capability of their own software processes, as well as to define areas and priorities for software process improvement. ISO 15504 also provides a roadmap for subsequent software process improvement. In the case of assessors, ISO 15504 provides a framework for conducting the actual assessments themselves. This is an extremely valuable and highly applicable approach to ensuring secure software development, sustainment and acquisition process, since it provides individual ratings of process capability. The organization will find itself in a position to identify individual processes that have insufficient maturity to ensure that software will be developed, sustained, or acquired in a secure fashion.

Like CMMI, ISO 15504 focuses on internal improvement rather than certification. Nevertheless, with the addition of a standard assessment process 15504 can be used to underwrite customer confidence. The assessment would encourage a culture of constant improvement and identify the actions that need to be taken to ensure a secure product. This will allow the organization to deploy the proper mechanisms to support and maintain a culture of secure software assurance and to engineer explicit processes to meet business requirements and optimize resources. The ability to understand and improve the capability of individual processes also ensures a capable organization with maximum responsiveness to customer and market requirements. That situation will minimize the full life-cycle costs and as a result maximize end-user satisfaction.

6.4.1 Assessment and the Secure Life Cycle

ISO 15504 defines a framework of base practices that underwrite the organization's self-assessment of its processes. That fact differentiates 15504 from quality assurance models such as ISO 9000. The 15504 assessment model evaluates the complete set of circumstances surrounding a given activity and constructs a Process Rating Profile rather than a simple pass/fail result (i.e., it is not like a CMMI staged assessment in that respect). However, ISO 15504 embodies an approach to assessing the adequacy of corporate practice compared with its stated goals, which is embodied directly in the ISO 9000 quality system assessments.

ISO 15504 also supports a corporate-level process capability determination by evaluating the proposed capability of the corporation’s selected process against a target capability profile. This is usually done to identify the inherent risks of future initiatives. However, it can also be used to understand the inherent risks in maintaining a given level of practice with respect to secure software work, which makes it an extremely valuable information gathering function for upper management. Levels of secure practice can be set for each process and then assessed to ensure that they have been met. The proposed levels of capability may be based on lessons learned from previous process assessments, or it might be grounded in an assessment done specifically to determine that a given process has achieved the right level of assurance.

Processes can, and probably do, exist at different levels of capability within any organization. Therefore, the organization needs to understand the capability of each of the individual processes that underwrite the development, sustainment, and acquisition of its software. It can then use that process-specific information to dictate the steps that will be taken to ensure each process meets basic requirements for secure practice. The priority and sequence of the improvement in those processes is always determined by the business case.

ISO 15504 employs common features to describe the management capability of each given process. Just as with the CMMI models, the common features are those management practices that affect an aspect of the implementation or institutionalization of a given process. For example, the Planned-and-Tracked Level of 15504 contains the Planning Performance, Disciplined Performance, Verifying Performance, and Tracking Performance common features. The Tracking Performance common feature consists of practices that track the status of the process using measurement and then implements corrective action as appropriate.

Thus, the common features and capability levels underwrite the assessment and improvement of an organization’s process capability. It is possible to assess capability by determining whether the generic management practices that constitute the common features are present in the process being assessed. In order to characterize the level of maturity, that complete set of practices is ordered by capability level. Thus identifying and characterizing the management practices that are present in an individual process allows the organization to determine the exact capability of all of the different processes within the scope of the assessment. What is lacking in the 15504 model is a validated set of common features that describe a secure process. However, it is likely that the current set of common features might be proven sufficient to characterize both a quality process and a secure one. That is the assumption of the ISO/IEC 21827 standard, which we will discuss in the next section.

In the case of improvement, the assessment provides an improvement route map because it provides explicit advice about how practices have to be added or changed in order to enhance a specific process to the desired level of capability. Just as in CMMI staged, the 15504 capability levels are each assessed based on the presence or absence of required common features (except for level 0, Not Performed) and these common features, in turn, can be broken down into a defined set of generic practices. By ensuring that those practices are being followed, the organization can rationally enhance and refine its process capability. Nevertheless, it has to be understood that the sophistication and complexity required for a given process to be secure depends on its context. For instance, the planning that guides conventional 5-person project teams will be much less resource intensive than the planning that goes into managing a 55-person team. This contextual requirement has to be kept in mind when assigning process adequacy ratings.

Process adequacy ratings are a unique aspect of 15504, and they can be extremely valuable to people interested in ensuring a secure software process. That is because an entire software operation can be described in individual detail using a process-rating framework. That framework is always aimed at evaluating a specific process instance. A process instance is a singular occurrence of a process that is uniquely identifiable and about which information can be gathered in a way that provides repeatable ratings.

Each process is characterized by a set of five process capability ratings, each of which is an aggregation of the practice adequacy ratings that belong to that level. Practice adequacy is a rating of the extent to which a practice meets its purpose as defined by the business case needs. These needs typically include the identified strengths, weaknesses, and risks inherent in the operation of the process. This helps to decide whether the processes are effective in achieving security goals. It also helps in identifying significant causes of defects, or time and cost overruns. That can be used to set priorities for improvement to the process.

Each process has a purpose and consists of a set of practices that are considered to implement that purpose. For example, the discrete ISO 15504 process “Develop software design” (ENG.3) contains this purpose statement: “The purpose of the develop software design process is to establish a software design that effectively accommodates the software requirements; at the top-level, this identifies the major software components and refines these into lower level software units which can be coded, compiled, and tested” [ISO 1998a].

That purpose statement indicates the reason for undertaking the “Develop software design” process. But it does not provide a specification of what has to be done in order to satisfy that purpose. That description is left to the next level down in the ISO 15504 descriptive hierarchy. That level is called the “Base Practices.”

For instance, “Develop software design” ENG.3 consists of the following set of base practices, all of which must be present in order for the requirements of its purpose to be satisfied:

ENG.3.1 - Develop software architectural design

ENG.3.2 - Design interfaces at top level

ENG.3.3 - Develop detailed design

ENG.3.4 - Establish traceability

The base practices that underlie each of the processes in the ISO 15504 model represent the unique, functional attributes of that process. These practices have to be present, even if their performance is not systematic. Or in practical terms, the base practice may be done ad hoc, be totally unpredictable, inconsistent, poorly planned, and/or result in poor quality products. Thus, the presence of a base practice within a process may be of minimal value and represent only the first step in building process capability. Therefore, the next step in the assessment is to determine the level of capability of each base practice.

6.4.2 ISO 15504 Capability Levels

Although the base practices have to be present, in reality the architectural components that are used to judge the capability of a given process are the common features. As we said earlier, these are the generic management practices, which are employed by each process. ISO groups these management practices in terms of how much they contribute to process maturity. In that respect, they function in the same capacity as the capability levels of the CMMI staged.

A common set of capability levels for judging process capability provides two distinct benefits. First, it helps the organization identify the specific sequence of management practices that it has to perform to achieve a given level of maturity for each base practice. This understanding is based on the implementation of a rational sequence of capabilities, which essentially serves as a roadmap for achieving a desired level of capability. For example, in order for a practice to reach the Well-Defined level (level 3), it is necessary to implement another process first, namely, “Define the process,” which is a level 2 activity.

As in CMMI staged, each grouping of common features provides a major enhancement in capability to that of its predecessors in the performance of a process. Successively implementing the requisite management practices for each level provides a rational way of progressing through the logical stages of the maturity process. The ISO 15504 capability levels are as follows [ISO 1998a].

Level 0: Not Performed: Base practices and common features do not exist at the “Not Performed” level. There are no easily identifiable work products or outputs of the process. Organizations functioning at this level are easily exploited because their software generally contains defects.

Level 1: Performed-Informally: At this level, there are requisite base practices present. However, the performance of these base practices may not be rigorously planned and tracked. Performance of the base practices depends on individual knowledge and effort. There are identifiable work products for the process. These work products can be used to document that performance. Individuals within the organization recognize that basic practices should be performed, and there is general agreement that these practices should be performed as required. This level is much more secure than level 0 because the organization as a whole is attempting to follow secure practices. However there is no control over the process, nor is there uniformity in how it is executed. Consequently, some products can be defect free while others can be easily exploited. Thus, it is impossible to tell which products are secure and which ones aren’t.

Level 2: Planned-and-Tracked: Base practices in the process are all planned and tracked. Moreover, that performance is verified based on the procedures, which are specified in the work plan. Work products conform to specified standards and requirements. The primary distinction between the Performed-Informally Level and the Planned and Tracked level is that the performance of the process is managed as a well-defined process. Organizations functioning at this level will consistently produce generally reliable software. However, the overall software process cannot be guaranteed to be conformant with best practice.

Level 3: Well-Defined: Base practices are carried out according to a well-defined process using accepted, tailored versions of standard, documented best practices. The primary distinction between the Planned-and-Tracked level and the Well-Defined level is that the processes at this level are managed through an organization-wide standard process. Organizations functioning at this level will produce software that is consistently at the highest standard of security because the product has been produced within a generally accepted best practice process. However, organizations at this level are not as adaptable to change as they should be, and so it is necessary to move to the next level in order to ensure that those practices are as up-to-date as necessary.

Level 4: Quantitatively Controlled: Organizations at this level employ detailed measures of performance to ensure the highest level of capability. The emphasis on quantitative information leads to a quantitative understanding of process capability and an improved ability to predict performance. That performance is managed based on organizationally standard data. The quality of work products is quantitatively characterized. The primary distinction between this level and the Well-Defined level is that the well-defined process can be adjusted and controlled with certainty because its performance is characterized by data and any defects are immediately identified and mitigated. Software produced by such an organization would be at the highest level of reliability because the exact status of all products is known and all defects are immediately remediated.

Level 5: Continuously Improving: Organizations at this level set quantitative process effectiveness and efficiency goals based on the business case. Continuous process improvement against these goals is enabled by quantitative feedback derived from performing the defined processes and from piloting innovative ideas and technologies. The primary distinction between this level and the Quantitatively Controlled level is that the defined process and the standard process undergo continuous refinement and improvement based on assessed understanding of the impact of change as it occurs. Consequently, software produced by an organization functioning at this level can be assumed to be at the highest level of reliability.

6.5 Adapting the ISO/IEC 21287 Standard Approach to Secure Software Assurance

In its general form and structure, the ISO/IEC 21287 standard is a mirror image of the 15504 model. However, the base practices of the ISO/IEC 21287 standard are specifically aligned to ensuring a secure process. Therefore, ISO 21287 defines the practices required to do good software security engineering. Those practices comprise a complete set of activities that extend over the entire life cycle of a trusted product or secure system. Like 15504, each of these activities is characterized by a goal statement that specifies its unique purpose. In addition, the goal statement also specifies the precise tasks to be performed in order to produce a correct work product. The 11 general process areas related to secure software production are

1. Administer Security Controls
2. Assess Impact
3. Assess Security Risk
4. Assess Threat
5. Assess Vulnerability
6. Build Assurance Argument
7. Coordinate Security
8. Monitor Security Posture
9. Provide Security Input

10. Specify Security Needs
11. Verify and Validate Security

The ISO/IEC 21287 standard also includes a set of 11 processes related to good project and program management:

1. Ensure Quality
2. Manage Configuration
3. Manage Project Risk
4. Monitor and Control Technical Effort
5. Plan Technical Effort
6. Define Organization’s Systems Engineering Process
7. Improve Organization’s Systems Engineering Process
8. Manage Product Line Evolution
9. Manage Systems Engineering Support Environment
10. Provide Ongoing Skills and Knowledge
11. Coordinate with Suppliers

In the ISO/IEC 21287 standard, the level of capability of each of these processes is characterized through a standard set of management attributes, which are applicable to any process. In effect, like 15504 these common attributes characterize the generic management capability of any assessed process. Improving the security engineering process would then involve increasing the generic level of management practice capability while ensuring that all the requisite base practices in the list (above) are performed.

The ISO/IEC 21287 capability dimension is structured in the same fashion as the ISO 15504 standard that we just discussed, and all of the assumptions about each of the levels just itemized (for 15504) also apply here. The five maturity levels in the ISO/IEC 21287 standard indicate increasing capability for each base practice. The difference is that ISO/IEC 21287 targets security engineering concerns rather than conventional software development. Each of the practices within the process framework can be assessed at one of these five levels of capability. The external audit develops evidence that a particular process purpose has been achieved. In order to attest to an increase in capability, the activities at each new level must represent proof that the requisite management attributes are present.

As in 15504, process capability attributes are grouped by capability levels. The intention is to provide a rational way for an organization to characterize its current process capability and to suggest a set of specific steps that can be taken to improve it. The actual security engineering capability of a given process is characterized by performance of an increasingly mature set of activities, which serve as commonly accepted proxies for enhanced capability. Presence or absence of those “standard” activities can then be used to authenticate the process’s degree of reliable execution.

6.6 The Business Case for Certifying Trust

This section is based on the assumption that the security of a product is tied to the capability of the people who create it. Thus, since vulnerabilities arise from software defects, it is important to be able to ensure that software products are developed by organizations that produce software with the fewest defects, i.e., the most capable organizations. The problem is that the marketplace is essentially anonymous. So customers are left with only two options: (1) deal with the usual list of suspects, all of whom know that you are their captive, or (2) buy a “pig in a poke” and hope that it isn’t malicious.

Since there is almost no justification for the latter approach, the former is typically the method of choice, even though taking competitive pressure out of the bidding process potentially means higher prices and less service. The ideal, of course, would be to maximize competitive pressure by spreading the net as wide as possible. The problem is that it would require precise knowledge of the exact level of capability of each of the suppliers who swam into the net, and it is impossible to know the capabilities of every potential supplier given the global nature of the software business.

On the other hand, if the level of capability of every software organization could be evaluated, it would then be a simple matter of companies only dealing with the ones who had the requisite level of capability. A similar concept has been used for years in the quality universe, in the form of the ISO 9000 quality system registries. So there is no reason why registrations of security engineering capability cannot be kept. If that were the case, it would do two things. First, it would promote competitive advantage for all of the business that chose to document their capability. And second, in doing so those businesses would embody processes that would ensure their capable performance and the security of their products. Given the level of threats in cyberspace, that assurance would be an extremely valuable outcome.

6.6.1 Certification: Ensuring a Trusted Relationship with an Anonymous Partner

From a global business standpoint, a financially justifiable and generally accepted demonstration of capability would accomplish two valuable aims. First, buyers of software would have objective independent assurance that they could trust a supplier, even if that supplier were in another culture 3,000 miles away. Second, suppliers would be able to gauge their level of capability and then determine how to improve it. Those twin advantages would reduce uncertainties in the acquisition process by identifying the risks associated with a given contractor. Moreover, by identifying the inherent risks, standard certification of capability would also provide an objective basis for trading off business requirements and project cost against each vendor’s areas of weakness when contract time rolls around. Finally, given the interdependence between process capability and the quality of the product, certification would also represent the best means of putting appropriate risk controls in place for whoever was eventually selected.

Thus, the opportunity to manage risks while increasing competitive pressure in the bidding process makes a strong case for some form of universal certification of supplier process capability. More importantly, third-party certification supported by audit is almost the only way to ensure a trust relationship between anonymous acquirers and suppliers, particularly in a global marketplace. So the question is, “What would be the most effective basis for a supplier certification process?”

For any business, the ability to evaluate the security engineering capability of each process lies at the heart of establishing a trusted relationship. That is because it can be assumed that if the supplier organization is operating at a specific level of capability, then it is essentially trustworthy. In evaluating any business relationship, it is important to know the specific risks involved in dealing with any given supplier. Since defects are tied to capability, the ability to characterize risk requires an in-depth understanding of how adequately each bidder's security engineering processes match up with the project's software assurance requirements.

The ability to understand vendor competency based on a commonly accepted model of good practice helps to address these problems. The advantages of such an understanding should be clear. First, a standardized basis for assessing capability can characterize organizational security functioning for a supplier's internal management consumption. More importantly, however, any subsequent external audits would provide objective evidence that could be used to document the general capability of the organization's security engineering practices. Ideally, these assessments should be recorded in some centralized third-party repository such as ISO to serve as a basis for accrediting trusted vendors worldwide. Nonetheless, a common basis for doing business based on capability assessments would be a tremendous asset both to suppliers and buyers.

Also, given the trend toward multitiered outsourced arrangements in software development and services, there is another reason for doing capability certification. Using this method it would be possible to create a hierarchy of profiles that specify both the process capability requirements of prime contractors as well as any subcontractors. In that respect then, it is much easier to ensure that the supply chain that underwrites complex system development does not have any weak links.

The ideal situation would be a single umbrella assessment model that would underwrite trust among all customer and supplier organizations. Independent auditing of individual capability could then ensure that any company that wishes to acquire a software product can find a trustworthy supplier anywhere in the world. The key to that effort is the existence of a valid, commonly accepted model and the availability of third-party certification of the assessment. The determination of exactly what model is appropriate for this has been the central point of this discussion. Accordingly, if agreement can be obtained about a common model, it should be much easier for CIOs to sleep at night, because it will be possible to make much more informed decisions about whom each organization is dealing with. Standard-based assessment of organizational capability is not a cure-all. But as secure software requirements become more pressing and the software business becomes more global, this approach can provide a realistic, viable, and attractive means for ensuring against security disasters. Given all that has been said earlier, businesses require a utilitarian road map to support their decision making in a complex and risky environment. The models we have discussed in this section all represent a detailed and consistent framework, and all of them should be considered in any future discussions about managing the secure software assurance process.

7 Globalization

These days the term globalization implies not only a global market, but also a global supply chain. In order to reduce development costs, many IT activities, including software development, have been moved to regions where the cost of labor can be significantly lower. Of course, there can be a number of risks associated with doing this [Mead 2003], including quality, liability, and so on. We will discuss a number of different models for the movement of IT activities elsewhere as a means of cost reduction. However, in the process of doing this, we need to consider the impact on software security. One obvious question is whether software that is developed elsewhere can be viewed as equally secure as software that is developed in the country where it will be used. We will discuss the tradeoff between globalization of software development and its impact on the software assurance cost/benefit argument.

7.1 Outsourcing Models

Researchers have identified four distinct models for IT managers to choose from when deciding to outsource and outsource offshore [Shao 2007].

1. **Onshore In-Sourcing**. In the onshore in-sourcing model, all IT services are kept in house. Every IT function, from creating IT systems to maintaining those same systems, is performed by internal IT staff. This approach can potentially foster a sense of pride among internal IT workers because they get a chance to maintain the finished product from the perspective of performance optimization and preventive maintenance.
2. **Onshore Outsourcing**. In the second model, both clients and vendors are domestic. This model is the most used approach in outsourcing contracts. This strategy actually promotes job security for domestic IT workers. Since these domestic vendors focus on IT, the internal IT workers that work for the outsourcing customer will continue to work in the same locations with the same types of roles. In many cases higher career mobility will be achievable because the domestic service contractor has a higher ceiling for IT oriented higher level work. The only most common effect of onshore outsourcing is that IT workers are moved around as far as who is paying their salaries.
3. **Offshore In-Sourcing**. Organizations that follow the offshore in-sourcing model set up their IT offices in less developed countries such as India and capitalize on the lower labor costs. This model is most used by large-scale intercontinental companies which have the resources to carry foreign locations.
4. **Offshore Outsourcing**. In a recent conversation with a high-level IT manager at Chrysler Financial Services, he spoke about a recent business trip that he went on to India. He was amazed by how you could be walking through the streets of what appeared to be a third world country, and as you turned the corner you'd see an IT service company's campus employing thousands of IT workers. This is an example of the fourth model for outsourcing.

   Offshore outsourcing presents some of the negatives of outsourcing. Movement of IT jobs overseas to lower waged IT workers results in a sense of instability among onshore IT workers. This trend has motivated many U.S. IT workers to attempt to stop IT work from going overseas through union pressure and political activism. A similar migration of jobs to foreign

> countries happened with manufacturing jobs over the past 30 years because of the cost savings of producing material goods in less developed countries. Because the marketplace in manufacturing has been global for a while, the only way that companies can retain their competitive advantage is through offshoring this type of work. It seems that IT offshore outsourcing is following this trend.

In recent years, another variation has arisen as major outsourcing suppliers in India and elsewhere have themselves started outsourcing [Dolan 2006].

7.1.1 Another View of Outsourcing Options

Other researchers have identified different outsourcing options [Lacity 1993]. Three of those options are (1) body shop, (2) project management, and (3) total outsourcing. The body shop option is best for short-term demands, such as contracting programmers who will be managed by internal management. Project management is an option that allows the parent organization to outsource and staff the project itself. An example would be to contract out disaster recovery to a vendor. Total outsourcing puts the vendor in total control of a significant body of work.

7.2 Costs and Benefits of Offshoring

One of the big reasons for the growth in outsourcing is that there has been a paradigm shift in how senior executives view IT. Many senior executives have made the decision to focus resources on what they're in business to do—their *core competencies*. An example would be that GM builds cars, not ERP systems. Therefore they will outsource a large-scale ERP implementation to a consulting firm before absorbing that cost. IT is not seen as a core competency by senior executives in most organizations. Another reason is that from senior executives' perspective, IT has an unclear value. Because of this, they view IT as overhead, and although it is an essential cost, it is a cost that should be minimized.

Reducing operating expenses is a good reason for outsourcing, but before an organization outsources, it should consider all of the costs involved. They include wages, mailing costs, benefits packages, IT network support for global networks, career enhancement technical training, in-house office supplies, concentrated software equipment, management of travel, cell phone and hotel phone charges, and occupancy bills. When analyzing these costs, it should not be the totality of the cost that's scrutinized. For example, analyzing totals such as total wages or total hotel phone charges does not show the costs from the perspectives of activity or process analysis. In order to accomplish this, one must follow the following five-step process:

1. Determine all activities that make up the function.
2. Streamline activities that coincide (10-15, with no one activity representing more than 20 percent of the total cost).
3. Identify the activity drivers, such as
   - labor driver (percentage of person's time used by an activity)
   - non-labor driver (percentage of office space utilized or number of machine hours required for the activity)
4. Actuate the cost of the drivers.
5. Compose the list of costs related to each driver.

There are several benefits that contribute to the trend to outsource IT offshore. Storage area networks (SANs) and networking technologies have matured to a point where iterations of different routines can be executed and remotely managed. An example of this is that you can have a SAN sitting in a data center in the U.S. with a SAN administrator administering the hardware from India for a fraction of the cost of using a domestic administrator. Some components that must be present are that the SAN must be tied into the network (securely accessible via the web), and in some instances there is a need to have operators on hand, where the hardware resides, in case there is a need to perform a physical task (putting a tape into a machine, or unplugging a switch or cable). It is also advantageous to have skilled project managers in place to manage the migration and the maintenance of the global system. With all of these factors being present, U.S. companies will continue to capitalize on cheaper labor costs.

Another practical benefit to outsourcing is that in disaster recovery, cheaper labor costs can be used when mirroring mission-critical servers overseas. In enterprise environments where there are large-scale SANs, mirroring clustered environments can be cost effective and beneficial for disaster recovery. For example, in the case of a blackout, having data mirrored on servers overseas means there is a backup in place. The tradeoff obviously deals with the risk factor. If you have mission-critical data running on your SAN and it’s mirrored to a SAN in India, you must ensure that there is adequate security.

7.3 Project Management Issues

Certain project management issues are destined to arise in outsourcing ventures due to differences in the cultures of the outsourcing organization and the vendor. It is important for the outsourcing organization to place liaisons in the vendor country in project management roles. This establishes a bridge of communication between the outsourcing organization and the vendor employees. It has been proven that when individuals are able to tie a face to a name, their dialogue is a lot less convoluted.

According to the PMI and the PMBOK, there are distinct differences between functional project managers and regular project managers. In outsourcing, there should always be a functional project manager from the outsourcing organization on site in the vendor country. If that is impossible, ongoing dialogue should be maintained via telecommunications.

7.4 Location

In weighing the impact of an outsourcing decision on software assurance, certain aspects of geographic location should be carefully evaluated. Infrastructure, political climate, trade relations with the United States, the expertise and education of the labor pool, and security practices vary widely from country to country. An article on Forbes.com [DiCarlo 2003] details some of these for India, the Philippines, Russia, China, Canada, Mexico, and Ireland.

7.5 Possible Tradeoffs

Clearly, there can be significant reductions in development cost when software is developed outside the country of usage, provided that good development and management processes are being used. However, we need to consider whether there is a negative impact on software security, and whether the associated risks can be successfully mitigated. The concerns are well known and are not that different from the typical security concerns—physical security, insider threat, incorrect implementation resulting in security flaws, and so on. All of these can be successfully addressed, but the cost of doing so must be added into the cost/benefit argument. We might add that these risks exist regardless of whether outsourcing is used, but they may be easier to manage and mitigate if the developer of the contracted software is geographically close.

It helps if there is representation from the contracting organization on site. This allows review of physical security as well as ongoing software development process review to confirm that security concerns are being addressed. Frequent videoconferencing or other virtual meetings can also be effective in monitoring the security of developed software. It can be a little trickier to deal with insider threat. In order to do this, it will be necessary to review the processes used in clearing personnel to work on the project and practices such as whether development is done in a secure environment or whether development can be done in a different location from the primary development site (for example, at home), thus introducing further risk. Our recommendation is to itemize the security risks associated with external development, to have a strategy for mitigating each risk, and to add the cost of the mitigation strategies to the cost/benefit argument. This may drive up the cost of “building security in,” but the overall development cost may still be lower when outsourcing is used.

8 Organizational Development

8.1 Introduction: Adding a New Challenge to an Existing Problem

Software projects have always been a crapshoot, with the odds seriously stacked against the player. For instance, a recent Borland study found that approximately 33% of all projects are canceled prior to deployment, 75% of all projects are completed late, and nearly 50% lack originally scheduled features and functions [Borland 2005, pg. 4]. In addition, it has been well documented that depending upon project size between 25% and 60% of all projects will fail; where “failure” means that the project is canceled or grossly exceeds its schedule [Jones 2005a].

Worse, this is not exactly a new phenomenon. Throughout the 1990s, industry studies reported almost exactly the same outcomes. During that period, the average project exceeded its budget by 90% and its schedule by 120%, and fewer than half of the projects initiated during that time finished on time and on budget [Construx 1998]. Likewise, a similar study done by KPMG Pete Marwick found that 87% of failed projects exceeded their initial schedule estimates by 30% or more. At the same time, 56% exceeded their budget estimates by 30% or more and 45% failed to produce expected benefits [KPMG 1996].

The root case of this less than sterling track record lies in the nature of software itself. Try building something that is invisible or accurately documenting something whose form only exists in the mind’s eye of a customer and you will understand the problem. Software development involves translating a customer’s abstract ideas about functionality into tangible program behaviors. That makes it hard to ensure anything consistent and repeatable about the process or its outcomes. Given those conditions, it might seem miraculous that anything useful has ever been produced by the industry, but the problem is just getting started. Now the product *also has to be secure*.

When defects were just quality issues, the problem of buggy code had marketing and customer relations ramifications. Today, the right kind of defect, exploited by the wrong kind of adversary, can lead to a 9/11 style outcome. That is the reason why no matter what the current list of excuses for defects, the “buck” has to “stop” when it comes to producing secure software.

8.2 Maintaining the Minimum Organizational Capability to Ensure Secure Software

In practical application, it is hard to make the business case for secure software. That is because organizations are composed of people and those people have varying degrees of capability. Variation isn’t a problem if a particular level of performance isn’t required because the company can always just keep patching their mistakes. However, where a specific level of proficiency is necessary to ensure a given level of performance, staff capability is a serious issue.

Staff capability is a major concern for business, since it is almost impossible to maintain a specific level of proficiency where constant turnover is a given. In that case, it becomes very important to adopt a well-defined process for developing and then assuring the organization’s overall capability. Best practice is essential for secure software work, since it defines the proper way to perform a given task. However, all sorts of factors can influence how closely and consistently any particular worker will follow any given practice. As a result, a standard organizational process has to be instituted to ensure that all required best practices are executed as specified in the software assurance plan.

Creating that process is an organizational development issue. It is also a precondition for making the business case for software assurance. It is a given that the organization is only going to be as secure as the capabilities of its people. Therefore, any discussion about the costs and benefits of secure processes is pure speculation until the people who will carry them out can be assured to be capable and willing to follow proper practice.

The term *discipline* simply denotes that a practice is reliably performed. Software assurance requires disciplined practice because in order to ensure a consistently secure product, all of the right practices have to be executed, by all participants, at all times, in a coordinated fashion. Accordingly, disciplined practice is essential to ensure that all of the products that are produced are secure all the time.

8.3 Learning to Discipline Cats

In many cases, consistent performance of disciplined practices will ensure the general security of code. But those practices will also impose additional work requirements. Because it is more work, it cannot just be assumed that the people who do that work will naturally accept and follow those new additional requirements. Instead, it should be assumed that people within the software organization have to be consciously motivated to carry out additional security tasks.

Motivation is an important factor in the software assurance process. Motivation initiates, directs, and sustains all forms of human behavior. Motivation is the factor that ensures a person’s willingness to consistently execute a given task or achieve a specific goal, even if the performance of the task itself is personally inconvenient. It also dictates the level and persistence of a person’s commitment to the overall concept of secure software. Consequently, motivation is the factor that underwrites disciplined performance.

Motivation is typically geared to accountability. This accountability comes from the enforcement of appropriate-practice (not best-practice) policies. Appropriate-practice policies are developed and documented by the organization to guide the entire process by which the software is created. These policies are then monitored for compliance as part of the overall organizational accountability system. The accountability system then rewards appropriate actions and discourages the inappropriate ones. However, it is impossible to enforce accountability if all of the appropriate-practice policies are not known or understood. Therefore, the organization also has to ensure that all of its employees know what they are expected to do, as well as the consequences of non-compliance.

Being able to ensure that everybody in the organization understands his or her exact role, responsibility, and function is the single most critical requirement in ensuring that software is developed correctly. That is because, no matter how potentially correct the security practices might be, if the people responsible for following those practices do not understand what they are supposed to do, there is almost no chance that the resulting work products will be secure.

Therefore, every organization has to undertake a deliberate effort to maintain every worker’s up-to-date knowledge of his or her individual security duties and accountabilities. The need to have an organized function in place to ensure a continuous level of security knowledge is particularly essential in light of the fact that the workforce in most businesses is constantly changing. As trained workers leave, or change jobs, and untrained people are added, there has to be a consistent effort to maintain a requisite level of knowledge and understanding.

Consequently, besides perfecting the technical end of the software assurance process, another aim of the software assurance function has to be to make certain that the function that ensures that understanding operates as intended. The means that most organizations employ to meet that obligation are *awareness*, *training*, and *education*.

8.4 Ensuring That Everybody in the Operation Is Knowledgeable

In ordinary use, the combination of awareness, training, and education is often called an AT&E program. But each of these delivery models represents a different approach to learning. Each has a distinct application and each is characterized by progressively more rigorous and extensive learning requirements. Because of that progression, these approaches are normally rolled out in practical application as a hierarchy.

At the basic level, which is awareness, the purpose of the learning is very broad, but the learning requirements themselves are limited. The next level up, which is training, builds on the awareness function. However, the application of training is restricted to fewer people and the learning is more in depth. Finally, at the top of the hierarchy, which is education, the application might be limited to a few key people, but the learning requirements are very broad and in depth. Some fundamentals on the subject of training and personnel development can be found in Appendix E.

8.4.1 Awareness Programs

Awareness is the lowest rung in the ladder. Effective awareness programs ensure that all employees at every level in the organization appreciate the need for, and are capable of executing, disciplined secure software practice, in a coordinated manner. This meets basic software assurance aims. However, the requirement for awareness varies across the organization. Awareness at the highest levels of the corporation sets the “tone at the top.” So, awareness programs at the executive level are focused on ensuring the strategic policy awareness issues facing the organization, as well as the costs, benefits, and overall implications of security.

At all of the other levels, it is necessary to maintain a relatively high degree of awareness of relevant software assurance practices. Therefore, everybody in the organization must be made aware of the specific security requirements that apply to their position. In addition, they have to be motivated to practice security in a disciplined fashion. Thus, a good awareness program will

- strengthen motivation—the program must motivate all users to practice security
- ensure effective focus—the program must concentrate on relevant and appropriate topics
- maintain participant interest—the program must ensure that individual participants will continue to be interested in security
- underwrite capable performance—the program must ensure effective security
- integrate the content—the program must ensure the full integration of the proper set of practices

However, awareness alone does not ensure reliable software assurance practice. It is also necessary to ensure that individuals responsible for executing specific assurance functions, such as static tests and inspections, are knowledgeable in the precise requirements of their role. That implies the need for a greater degree of knowledge and capability than is typically provided by an awareness function. This is typically underwritten by formal training.

8.4.2 Training Programs

Training is organized instruction that is intended to produce an explicit outcome. Consequently, it emphasizes job-specific skills. The purpose of training is to make sure that organizational functions, which are required to ensure safe and secure software, are performed correctly. Training ensures that all participants in the process have the specific skills necessary to carry out their assignments and that the level of organizational capability is continuously maintained. Training can be expensive, but it is an effective way to guarantee capable long-term execution of software assurance processes.

Nonetheless, because it is based on skills rather than concepts, training is too narrow to ensure that the software assurance process itself is executed correctly across the entire organization. Instead, training prepares individual workers to execute a series of steps without concern for the context or the reasons why the steps might be necessary. Training provides a quick and satisfactory outcome if the known threats to software never change or if adaptation to new threats is not required. However, most assurance situations are dynamic and complex. Therefore, training does not provide the overall strategic understanding that is necessary to establish a lasting security solution. A program of formal education is required to ensure that the organization’s code is maintained continuously secure.

8.4.3 Education Programs

Education is oriented toward knowledge acquisition, rather than the development of short-term skills. It ensures an intelligent, rather than rote, response. It establishes understanding of the principles of secure software development as well as the critical thinking abilities that will be needed to evolve the software development process through a continually changing and uncertain threatscape. For that reason, the few individuals in the organization who are responsible for the long-term guidance of the security function must undergo formal and in-depth education in software assurance principles and practices.

Education can be distinguished from training by its scope, as well as the intent of the learning process. In a training environment, the employee acquires skills as part of a defined set of job criteria. In an educational context, the employee is taught to think more about the implications of what he or she is learning. The learner must be able to analyze, evaluate, and then select the optimum security response from all alternatives. Thus learners are encouraged to critically examine and evaluate the problem and to respond appropriately by tailoring fundamental principles into a solution that precisely fits the situation.

The practical aim of education is to develop the ability to integrate new knowledge and skills into day-to-day security practice. The specific outcome of an institutionalized education process is the ability of executives, managers, and workers to adapt to new situations as they arise. Given what has been said about the constantly changing nature of threats and vulnerabilities, this is an essential survival skill for the leadership of any organization.

8.5 Increasing Organizational Capability Through AT&E

The outcome of a properly administered AT&E program is an increased level of organizational capability. This is a strategic concept. It is based on the achievement of five progressively more capable states of security:

- Recognition—the organization recognizes the need for security.
- Informal Realization—the organization understands informal security practices.
- Security Understanding—the security practices are planned and monitored.
- Deliberate Control—decisions about security practices are based on data.
- Continuous Adaptation—practices adapt to changes and are continuously improving.

The levels of capability are progressively achieved through targeted awareness, training, and education processes.

8.5.1 Security Recognition

The most fundamental level is simple Recognition. Here, the majority of the participants are able to recognize that secure software is a valid and necessary concern. Until that fundamental state of recognition is achieved, the organization is essentially operating without any concept of secure practice. Once adequate recognition is established, however, individual members begin to understand that exploitation of coding flaws is a concern. This may not necessarily be in any deliberate or actively organized fashion, but it does involve a persistent underlying appreciation that security practice is necessary.

8.5.2 Informal Realization

At the next level, Informal Realization, members of the organization become more conscious of the need to ensure against software defects. Every worker is aware that those concerns exist. Workers might also follow rudimentary assurance procedures in response to that understanding. Thus, this level is supported by a more involved awareness program.

The awareness program that underlies informal realization presents security issues that have been expressly identified as concerns, such as buffer overflows. It might also present general practices to address these concerns, such as parameter checking. This is done on an ad hoc or informational basis. The best practices that are designed to avoid common coding errors are not sufficiently specific and their performance is not organized well enough to ensure that security is embedded in the standard operation. That happens in the next step.

8.5.3 Security Understanding

The third stage, Security Understanding, is the first level where a consciously planned and formal security effort takes place. At this stage, the organization understands and acts on a commonly accepted understanding of the need for some form of formal security practice. The response might not be extensive and it is often dependent on individual willingness, but it is recognizable in that standard software assurance procedures are planned and documented in a systematic fashion.

The fact that security procedures have been formally documented allows the organization to implement a training program. Training is typically done to enforce understanding of the requisite security practices that are associated with each generic role. For instance, there might be targeted programs for executives regarding the business consequences of exploitation, a different one for managers aimed at implementing monitoring and control functions, and another for workers aimed at ensuring that best practices are followed.

The worker training programs might be subdivided by operation, such as development, versus acquisition, versus sustainment. The aim of each program though is to foster understanding of the security procedures that are appropriate to that role or function. These programs are generally not oriented toward ensuring specific skills beyond the understanding of the security practices that are required to carry out basic work. That is done in the next stage.

8.5.4 Deliberate Control

The fourth stage, Deliberate Control, is typical of a well-organized software assurance operation. Deliberate control is characterized by an institutionalized software assurance response that is built around providing a tailored set of skills for each relevant position. These skills are defined and managed based on a precise knowledge of the requirements of each individual’s role in the organization.

The execution of these security tasks is monitored using quantitative measures of performance, such as defect density. Deliberate control is enforced by defined accountability. Because it is objectively monitored, the security operation is fully managed by the organization’s top-level executive team. At this level of functioning, the organization can be considered both safe from common threats and actively practicing the steps that are necessary to maintain that requisite level of security.

This state comprises a targeted mix of training and education. Coordination and administration of the program is designed to achieve specific assurance outcomes. The training and education program communicates the precise knowledge and skills that are needed to correctly perform specific security practices that are required by each function. This is reinforced through periodic retraining.

Training at this level is a carefully planned activity that requires many of the activities performed by the personnel security function, such as job definition, job classification, and privilege setting, to make it successful. The outcome provides a very high level of carefully controlled assurance. However, this is not yet the highest level of education possible.

8.5.5 Continuous Adaptation

At this final and fifth level, the software assurance function is fully optimizing. It not only carries out all of the practices necessary to ensure secure code within the dictates of the situation, but it continues to evolve those practices as conditions change. Organizations at this level are capable of adapting to new threats as they arise. That allows them to maintain consistently effective software assurance countermeasures as well as an active response to any new threat. They are safe from harm because they are protected from all but the most unforeseen events, and they are capable of a rapid and meaningful reaction to any threat that might occur.

This stage is achieved by ensuring that workers master the critical thinking skills necessary to identify and solve problems. That requires a high level of knowledge of the elements and requirements of the field, as well as the thought processes to allow people to adapt these principles to new situations as they arise.

The classic mechanism for reaching this level of competence is a well-designed educational program. Skill training might also be among the factors needed to achieve this level. Nevertheless, the integration of that knowledge into the capability to respond correctly to new or unanticipated events falls within the realm of education.

8.6 The Soft Side of Organizational Development

It is recognized that many of the security problems in our complex world can best be addressed by taking an inter-disciplinary approach. In this section, it is argued that the social sciences have a valuable role to play when helping build business cases for information security and software assurance. This is because understanding these soft human issues will support better decision making in an organization.

What drives an organization forward is often encapsulated in its mission and vision statements. In creating a business case it stands to reason the case stands a better chance of being supported if it aligns with both these statements. Taking this a step further, understanding the motives and emotions behind these two statements in terms of the decision makers would help shape the content and help position the arguments more effectively. For example, if the vision statement is expansionist, then the risk appetite is likely to be greater than if the vision statement is consolidatory.

There are also important human-related learning opportunities when looking for effective ways of presenting and communicating the business case. The importance of effective communication and positioning of the business case cannot be over emphasized, and many large organizations have clear rules in drafting business cases to make the best use of the senior manager’s time. Clear use of language, diagrams, examples, videos, and demonstrations all help to get the key messages across.

Social science findings which could provide support for the business case are described in Appendix F.

8.7 Some General Conclusions

A formal and well-run AT&E program is a critically important advantage for a software organization because, in the end, no matter how well-intentioned your staff might be, without sufficient knowledge in secure coding practice, your assurance capability will be limited.

The type of dynamic approach outlined here is an ongoing commitment. Therefore, it cannot be stressed enough that the organizational entity that is given the responsibility for training must constantly monitor and control the development of the program and the personnel resource through formal assessment and review.

The maturation of an AT&E program is a continuous activity that flows from the refinement of security knowledge as well as new knowledge gained through performance of security activities. The training operation requires a total commitment by the organization, particularly the top-level people, to maintaining a dynamic and complete understanding of all necessary requirements and capabilities. This is essential in order to develop the programmatic responses required to meet the demands of an evolving threatscape. Nonetheless, if this dictate is adhered to, AT&E can provide the operational backbone necessary to ensure that the organization will stay secure.

9 Case Studies and Examples

9.1 Background

No argument is more compelling than actual case studies that illustrate the benefit of paying attention to security issues when building newly developed software. Although many organizations have just started to collect the needed data, we have already identified some excellent examples.

The case studies and examples presented here include

- information from a large corporation that was building security into their products
- interesting information from the SAFECode group at the Making the Business Case for Software Assurance Workshop [SEI 2008]
- Microsoft data resulting from use of the security development lifecycle [Lipner 2005] presented at a Software Assurance Forum
- a case study presented by Fortify at a Software Assurance Forum
- data collected over a period of time by the COCOMO group

9.2 Case Studies and Examples

9.2.1 Case 1: Large Corporation

We had some interesting discussions with executives at a large corporation. This company was addressing security during the development of selected critical software. It became clear, however, that the software was deemed critical for reasons that can be summarized as follows:

- The corporation believed the software provided it a competitive advantage in the marketplace.
- Risk analysis activities identified the software as critical to the business.
- Risk analysis reviews identified the software as high risk, with possibly severe consequences to the business or the company’s reputation if attacks or break-ins were to occur.

9.2.2 Case 2: SAFECode

The Making the Business Case for Software Assurance Workshop included participation from members of the SAFECode group.[23] SAFECode member organizations include Microsoft, Symantec, Juniper Networks, SAP, EMC Corporation, and Nokia. The SAFECode members discussed the reasons why some organizations had an increased focus on security. For instance Dan Reddy, of EMC Corporation, indicated that his market *demanded* a focus on security. SAFECode members Wes Higaki, of Symantec, and Michael Howard, of Microsoft, have since presented SAFECode’s work at the October 2008 Software Assurance Forum [Higaki 2008]. SAFECode has also published a set of best practices to support its work [SAFECode 2008]. The SAFECode objectives are as follows:

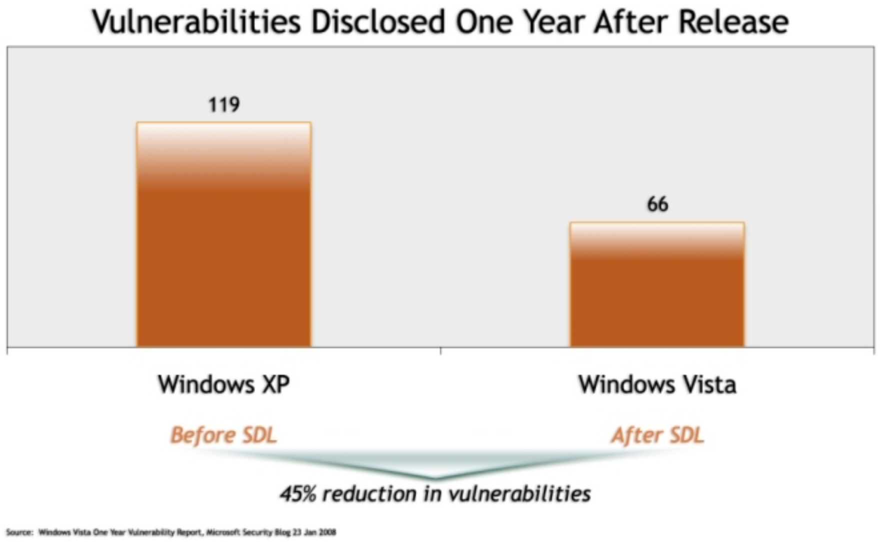
[23] http://www.safecode.org/

- Increase understanding of the secure development methods and integrity controls used by vendors.
- Promote proven software assurance practices among vendors and customers to foster a more trusted ecosystem.
- Identify opportunities to leverage vendor software assurance practices to better manage enterprise risks.
- Foster essential university curriculum changes needed to support the cyber ecosystem.
- Catalyze action on key research and development initiatives in the area of software assurance.

SAFECode's presentation at the Software Assurance Forum indicated that "vendors who have implemented these best practices have seen dramatic improvements in software product assurance and security" [SAFECode 2008].

9.2.3 Case 3: Microsoft

Microsoft data presented by Steve Lipner at a Software Assurance Forum had previously indicated a significant reduction in patch levels (number of vulnerabilities) as a result of the Microsoft security "push." The Microsoft data, which cover a recent five-year period, indicated that patch levels were reduced by half in versions of Windows software developed after the security push. Figure 2 presents the positive effect of the security push on Microsoft Windows and Vista.[24] Note that this particular example illustrates the benefits without discussion of the cost, but presumably Microsoft finds that this is worthwhile or they would not be doing it.



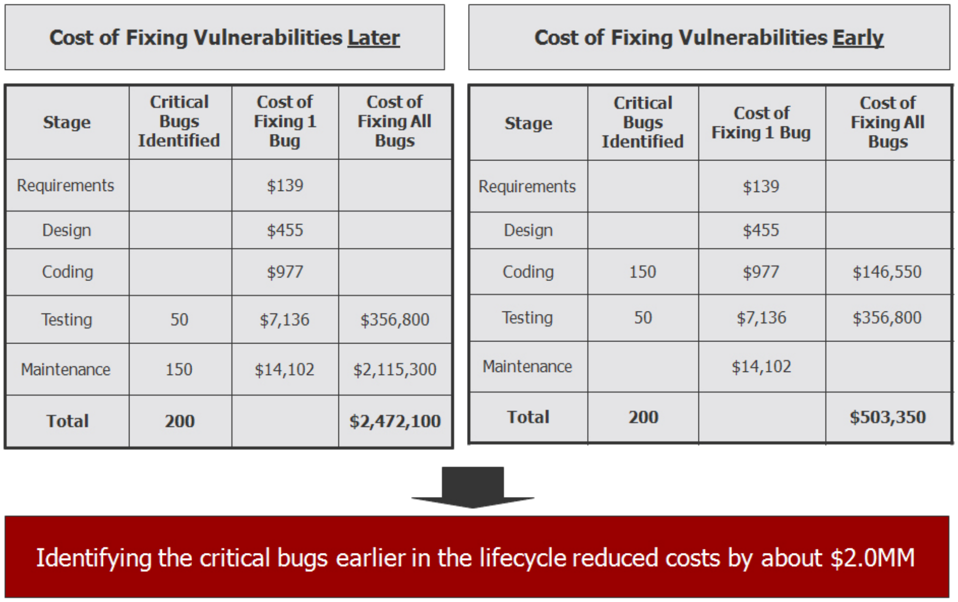
Source: Windows Vista One Year Vulnerability Report, Microsoft Security Blog 23 Jan 2008

Figure 2: Effect of Microsoft Security "Push" on Windows and Vista

[24] Figure 2 copyright 2009 Microsoft Corporation. For additional data related to the effect of the Microsoft security push on other Microsoft products, visit http://msdn.microsoft.com/en-us/security/cc424866.aspx [Microsoft 2009].

9.2.4 Case 4: Fortify Case Study Data

In October 2008, at the DHS Software Assurance Forum, Barmak Meftah of Fortify presented case study data based on actual client data. The Fortify data definitely showed the powerful cost/benefit case for building security in. The Fortify data suggested that it cost 100 times more to fix a single vulnerability in operational software than to fix it at the requirements stage. This data, presented in Figure 3, is derived from examination of a software vendor's products [Meftah 2008].



Cost of Fixing Vulnerabilities Later

Stage	Critical Bugs Identified	Cost of Fixing 1 Bug	Cost of Fixing All Bugs
Requirements		$139	
Design		$455	
Coding		$977	
Testing	50	$7,136	$356,800
Maintenance	150	$14,102	$2,115,300
Total	**200**		**$2,472,100**

Cost of Fixing Vulnerabilities Early

Stage	Critical Bugs Identified	Cost of Fixing 1 Bug	Cost of Fixing All Bugs
Requirements		$139	
Design		$455	
Coding	150	$977	$146,550
Testing	50	$7,136	$356,800
Maintenance		$14,102	
Total	**200**		**$503,350**

Identifying the critical bugs earlier in the lifecycle reduced costs by about $2.0MM

Figure 3: Fortify Case Study Data

9.2.5 Case 5: COCOMO data

Data collected on errors by Barry Boehm also suggest a ratio that could be as high as 100 to 1 for fixing errors late in the life cycle, but it can also be as low as 5 to 1 [Boehm 2001]. On average, Boehm's is not as dramatic as the data noted above, but nevertheless supports the cost/benefit argument. Note that this data covers the more general case of fixing all software errors, whereas the Fortify data is specific to vulnerabilities. Either way, the data support the fact that it is much less costly to fix errors and vulnerabilities early in software system development rather than after the software has become operational.

9.3 Conclusion

The business case for building software assurance in during software development is made in a variety of ways. It can be based on risk analysis results, competitive advantage, perceived damage to reputation from successful attacks, and actual cost/benefit data. Although we don't yet have many case studies to support the business case, the case studies and associated data that we have identified are very compelling.

10 Conclusion and Recommendations

10.1 Getting Started

We've provided a lot of information and guidance. By now it should have become clear that making the business case for software assurance is not simply a matter of doing a few calculations. There is a larger context that needs to be considered, as is usually the case, if real change is to occur. The following steps are recommended in order to effectively make the business case for software assurance and to effect the changes that go along with it.

1. **Obtain executive management support**. It's almost impossible to make the changes that are needed to support the business case for software assurance without management support at some level. At a minimum, support is needed to try to improve things on a few pilot projects.
2. **Consider the environment in which you operate**. Does globalization affect you? Are there specific regulations or standards that must be considered? These questions can influence the way you tackle this problem.
3. **Provide the necessary training**. One of the significant elements of the Microsoft security "push" and other corporate programs, such as IBM's software engineering education program, is a commitment to provide the needed training. The appropriate people in the organization need to understand what it is you are trying to do, why, and how to do it.
4. **Commit to and achieve an appropriate level of software process improvement**. Regardless of the process you use, some sort of codified software development process is needed in order to provide a framework for the changes you are trying to effect.
5. **Perform a risk assessment**. If you are going to make the business case for software assurance, you need to understand your current level of risk and prioritize the risks that you will tackle.
6. **Decide what you will measure**. If you are going to have any evidence of cost/benefit, you will need to have a way of measuring the results. This may involve use of some of the models that were discussed earlier, development of your own measures of interest, or use of data that you are already collecting.
7. **Implement the approach on selected projects**. Go ahead and collect the needed data to assess whether there really is a valid cost/benefit argument to be made for software assurance. The case studies that we presented are the result of such implementations.
8. **Provide feedback for improvement**. An effort such as this is never intended to be a one-time effort. If your cost/benefit experiments are successful, see how they can become part of your standard practices. Assess whether you can collect and evaluate data more efficiently. Assess whether you are collecting the right data. If your cost/benefit experiments are not successful (cost outweighs benefit), ask yourself why. Is it because software assurance is not a concern for your organization? Did you collect the wrong data? Were staff members not exposed to the needed training? Are you trying to do too much?

10.2 Conclusion

This guide provides an overview of a variety of topics related to making the business case for software assurance. Unfortunately, this is not a trivial undertaking, and there is no single cookbook method that is guaranteed to work in all cases. In order to achieve long-term success, organizational commitment will be needed. However, it is possible to start with a relatively small number of pilot projects that can provide the data needed to effect broader change. This guide and the associated references can help you get started along this worthwhile path.

Appendix A: The "Security" in Software Assurance

In presenting any business case, it must be made clear to decision makers what problem an investment is trying to fix, as well as the proposed solution. In doing this, clear, unambiguous language is key to a successful outcome. The following explanation of the definition of software assurance may help reconcile where security fits in.

Bell Labs researchers have conducted some interesting studies on the concept of intrinsic vulnerabilities [Rauscher 2006]. They state that in any infrastructure, which includes software, there are vulnerabilities that cannot be designed or built out in an economic sense and can only be mitigated against. These are *intrinsic* vulnerabilities. By contrast, there are vulnerabilities that *can* be designed and built out—*non-intrinsic* ones—and therefore doing so removes the need for mitigation.

Understanding the difference between intrinsic and non-intrinsic vulnerabilities is at the heart of good software assurance.

A solution for a non-intrinsic vulnerability might be the requirement to develop an authentication mechanism to protect against unauthorized use. By comparison, a solution for an intrinsic vulnerability might be installation of the system and its data on an isolated machine that has no internet connectivity.

It is useful to note that the first is an example of a *functional requirement*, while the second is an example of a *non-functional requirement*. Software assurance addresses both.

Reconciling this back to the original definition, software assurance is "a level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at any time during its life cycle." This part of the definition addresses the non-intrinsic vulnerabilities and the non-functional requirements caused by software flaws, whereas the rest of the definition, "and that the software functions in the intended manner," addresses the intrinsic vulnerabilities and the functional requirements for security.

Appendix B: Cost/Benefit Examples

The Build Security In website contains a number of articles with examples for computing cost/benefit for secure software:

- “Estimating Benefits from Investing in Secure Software Development” [Arora 2008] includes two calculators under the Cost and Benefit Calculators paragraph.
- “Calculating Security Return on Investment” [O’Neill 2007] includes worked examples.
- “Making Business-Based Security Investment Decisions – A Dashboard Approach” [Allen 2008a] includes a discussion on how to decide which security investments to make.
- “A Common Sense Way to Make the Business Case for Software Assurance [Bailey 2008b] shows a cost/benefit method of calculation using the Balanced Scorecard approach. This approach is described below, and is largely derived from the article on the site.

Here we demonstrate how a true cost/benefit for secure software can be derived using three generic practice areas: (1) threat/risk understanding, (2) implementation of security requirements, and (3) operational security testing. Having an accurate cost for these aspects of the software assurance process would allow decision makers to make intelligent decisions about the level of investment they wish to make.

In Section 2, we identified commonly accepted models for valuation of IT assets. One of the most widely used of these is the balanced scorecard [Kaplan 1992]. We will therefore use that approach to illustrate how a business case might be built for software assurance.

The balanced scorecard is particularly applicable to developing a cost/benefit argument for software security because it is designed to let companies assess the performance of a target process against factors that are outside of the scope of traditional time and effort costing. At its core, the scorecard seeks to equate the standard financial indicators of performance with more fluid indicators such as customer relationship, operational excellence, and the organization’s ability to learn and improve [Kaplan 1992]. Table 5 summarizes these categories and gives some examples of ways they can be measured.

Table 5: Categories of Measures for Four Perspectives of the Balanced Scorecard

Financial Perspective Measures	**Customer Perspective Measures**
• Net Income • Operating Margin • Economic Value Added • Revenue Growth	• Customer Satisfaction, External • Customer Satisfaction, Internal • Customer Loyalty

Operational Perspective Measures	Learn/Grow Perspective Measures
• Safety • Process Enhancement • Operational Efficiency • Productivity	• Employee Personal Development • Employee Satisfaction • Organizational Enhancement

We could envision additional measures, such as customer trust; however, Table 5 illustrates the balanced scorecard approach as described by the original authors. The balanced scorecard requires the use of metrics that have been specifically customized to an individual organization's particular environment. Generally there are three types of metrics involved. The first are the metrics used to describe internal technical functions. Examples of these are reliability and defect rate. These measures are not particularly useful to non-technical managers and upper level policy makers, but they are objective and easy to aggregate into information elements that can help technical managers assign value to the security aspect of the IT function.

The second category contains measures that form the ingredients of comparisons, or "report cards." These are intended for use by senior executives. For example, if software assurance is designated a cost center, the goal is to show how those costs have changed over time, or to assess how they compare with costs in similar companies. Examples of measures in this area include operation costs broken out on a per-user basis.

The final category includes metrics that are intended for use by the business side. These can include such things as utilization analyses and cost and budget projections. These measures allow an organization to estimate the business impact of a given activity. This evaluation is absolutely essential if it wants be able to prioritize the level of resources it is willing to commit to the assurance process for each item of its operational software.

Table 6 contains a non-exhaustive list of potential measures that might apply to a costing process for software assurance for each scorecard category. Using a tailored scorecard composed of standard metrics, the organization can then assign a quantitative value for each of its software assurance activities. And it can track the effectiveness of those activities using data obtained through one (or all) of these categories.

Table 6: Sample Set of Measures for Assigning Value to Software Assurance

Financial Perspective Measures	Customer Perspective Measures
Measures of Technical Value • Risk assessment/threat model activity cost • Operational test activity cost • Operational test defect removal cost • Risk assessment/threat model defect removal cost Measures of Comparative Value • TCO for software assurance activities	Measures of Technical Value • Customer initiated change • User and focus group feedback • Problem resolution reports Measures of Comparative Value • Risk assessment/threat model action items • Pen test benchmarking data

• Net overall income increase over time • Net overall revenue per unit Measures of Business Value • Economic value added for software assurance • Increase/decrease in cost per unit • Cash flow expressed as P&L	• CC capability certifications Measures of Business Value • Cost of customer initiated change • Cost to address unmet requirements • Cost of customer initiated rework
Operational Perspective Measures Measures of Technical Value • Exploitable defect identification rate • Exploitable defect removal percent • Exploitable defect density • Percent security requirements satisfied • Change requests tied to incidents • Operational resilience Measures of Comparative Value • Performance against budget • Productivity • Mean time to incident Measures of Business Value • Marginal increase/decrease in incidents • Marginal change in cost of incident • Risk assessment cost estimates • Growth in personnel capability	**Learn/Grow Perspective Measures** Measures of Technical Value • Assessed employee security capability • SwA training/development activities • Personal certifications (e.g., CISSP) Measures of Comparative Value • Growth in satisfaction with process • Growth in assessed security process • Growth in security/safety performance Measure of Business Value • Training performance indices • Employee performance indices • Security certifications per unit

It is possible to construct a meaningful business case for security from data elements such as these. For example, the organization could begin collection of data on all four perspectives, which it could equate to relevant indices. A sample set of these follows.

Financial Perspective Measures

Measures of Technical Value

- Risk assessment/threat model activity cost versus exploitable defect identification rate
- Risk assessment/threat model activity cost versus percent security requirements satisfied
- Operational test defect removal cost versus mean time to incident
- Risk assessment/threat model defect removal cost versus customer initiated change cost
- Exploitable defect removal percent versus productivity (e.g., in LOC)
- Exploitable defect density versus marginal increase/decrease in incidents

- Change requests tied to incidents versus performance against budget

Measures of Comparative Value

- TCO for software assurance activities versus net overall income increase over time
- Economic value added for SwA versus net overall revenue per unit
- Risk assessment/threat model action items versus cost to address unmet requirements
- Pen test benchmarking data versus cost of customer initiated change
- Assessed employee security capability versus growth in satisfaction with process
- CC capability certifications as a measure of growth in assessed security process

Measures of Business Value

- Operational test activity cost versus increase/decrease in cost per unit
- Growth in personnel capability versus marginal change in cost of incident
- Training performance indices versus cost of customer initiated rework
- SwA training/development activities versus personal certifications (e.g., CISSP)
- Security certifications per unit growth in unit security/safety performance

The indices cited are not a recommendation about the specific ones that an individual business might adopt. Instead this list is an attempt to illustrate two things. First, it is possible to collect quantitative data that can be used to prove, or even disprove, a business case for software assurance. Second, it demonstrates that data can be turned into meaningful information that can be easily understood by executive decision makers. The latter feature might be more important than the former, since any initiative such as this has to be valued and supported by upper management in order to succeed.

Results

The indices presented in the balanced scorecard example all have a quantitative basis that can provide the data points for any form of economic analysis. The same would hold true for all of the other approaches. For instance, cost-oriented models such as Total Cost of Ownership (TCO) and Economic Value Added (EVA) are driven by quantitative operational data. Needless to say, quantitative estimation models such as Real Options Valuation (ROV) also depend for accuracy on hard data derived from actual practice.

This example has demonstrated how costs and benefits of software assurance can be derived from a standard set of practices, which can be reliably related through the current literature to the assurance case and the assurance case alone. The ongoing operational information that this would generate, using whatever economic valuation approach that an organization might deem appropriate, would establish an exact business value for the assurance process.

Moreover, that information can then be used by decision makers to make intelligent decisions about the amount of investment that they wish to make in securing their software and information assets. The confidence that an organization can gain from this insight will allow them to optimize their software assurance investment and thereby improve the safety and security of society as a whole.

Appendix C: SIDD Examples

This appendix contains the following three sections in illustration of the Security Investment Decision Dashboard (SIDD) described in Section 3:

- C.1 Category and Indicator Rankings
- C.2 Scores for One Investment in One Category
- C.3 Summary Results for Four Investments

Please note that larger versions of the tables and charts in this appendix are available in the "Making Business-Based Security Investment Decisions – A Dashboard Approach" article [Allen 2008a].

C.1 Category and Indicator Rankings

In this example, the "Criticality and Risk" category is ranked as "1" and is the most important category for any organizational investment decision. "Measurability" is ranked as "7" and is thus the least important category-level criteria.

The indicator that has the highest priority here is the "Cost of NOT doing this investment, in terms of potential exposure and residual risk." It is ranked as "1" and is the most important indicator for any organizational investment decision. As you might expect, the three indicators under "Measurability" are the least important indicators.

	Category / Indicator	**Cat Rank (1-7)**	**Ind Rank (1-33)**
Category	**Cost**	2	
Consider	Estimated total cost to accomplish this investment, taking into account the potential cost savings and/or risk reduction to the organization		
Indicators	Overt cost in dollars at outset to accomplish this investment		6
	Estimated life-cycle cost in dollars over time to sustain this investment		7
	Cost of NOT doing this investment, in terms of potential exposure and residual risk (high = investment is more necessary)		1
	Potential cost savings to organization beyond breakeven point, if quantifiable (ROI), over time (high = better)		9
Category	**Criticality and Risk**	1	
Consider	Degree to which this investment contributes to meeting the organization's business objectives and risk management goals		
Indicators	Degree to which this investment is key or mainstream in helping the organization meet its primary objectives and critical success factors		4
	Degree of risk (as assessed in terms of likelihood and potential impact — high/medium/low priority) mitigated by this investment		3
	Degree to which this investment helps the organization protect stakeholders' (shareholders) interests		5

	Category / Indicator	Cat Rank (1-7)	Ind Rank (1-33)
Category	**Feasibility**	3	
Consider	Likelihood of investment success		
Indicators	Likelihood of success on first try		10
	Likelihood of success on subsequent tries (if first try fails)		11
	Likelihood that turnover among management and/or board of directors will negate work expended on this investment (low likelihood = better)		20
	Likelihood that this investment will need to be rolled back (low = better)		16
Category	**Positive Interdependencies**	6	
Consider	(1) Degree to which this investment integrates with or represents reasonable changes to existing organizational processes and practices, rather than requiring new ones		
	(2) Degree to which this investment paves the way for future investments (compliance, policy, risk management, etc.)		
Indicators	Degree to which other investments/tasks are dependent on this one (i.e., degree to which this investment makes it easier to accomplish additional tasks)		28
	Degree to which the accomplishment of this investment makes it easier for the organization to comply with current laws and regulations		2
	Degree to which the accomplishment of this investment makes it easier for the organization to comply with potential **new** laws and regulations in the future		29
	Degree to which existing knowledge and/or skills can be used to accomplish this investment, rather than requiring new skills/knowledge		25
	Degree to which this investment produces positive side effects (e.g., enhancing brand/reputation, building customer trust, benefiting supply chain partners)		27
Category	**Involvement**	5	
Consider	What level of involvement and buy-in are required from various parties for investment success — both within and outside of the organization		
Indicators	Level of buy-in required throughout the organization (Must all employees be on board 100% for this to work? Or only a subset, such as management and certain key employees?)		12
	Extent to which this investment requires the active involvement of many departments across the organization		21
	Number of people who need to be actively involved		24
	Level of involvement by third parties required (partners, consultants, vendors, etc.)		13
	Degree of external, independent assessment/auditing (vs. in-house assessment/auditing) required		30

Category	**Measurability**	7	
Consider	How measurable the outcome of this investment is		
Indicators	Degree to which this investment can be evaluated using existing approaches and reporting mechanisms		32
	Measurability of the outcome (Can it be quantified in tangible terms (revenue, market share, stock price, etc.)?		31
	If the outcome is intangible (e.g., goodwill, increased customer trust, enhanced brand), whether the benefits can be demonstrated against meaningful business success factors		33
Category	**Time and effort required**	4	
Consider	(1) Level of staff-hours required to accomplish this investment		
	(2) How long it will take to reach break-even cost for this investment		
Indicators	Board of directors time required		17
	Senior management time required		18
	Cross-organizational team/steering committee time required		19
	Middle and lower management time required		23
	Other key staff time required		22
	Time likely needed to achieve the required level of buy-in		14
	Time required to achieve first demonstrated results of task		26
	Time required to achieve full adoption and use of investment results across all affected business units		15
	Time to achieve breakeven, if quantifiable		8

C.2 Scores for One Investment in One Category

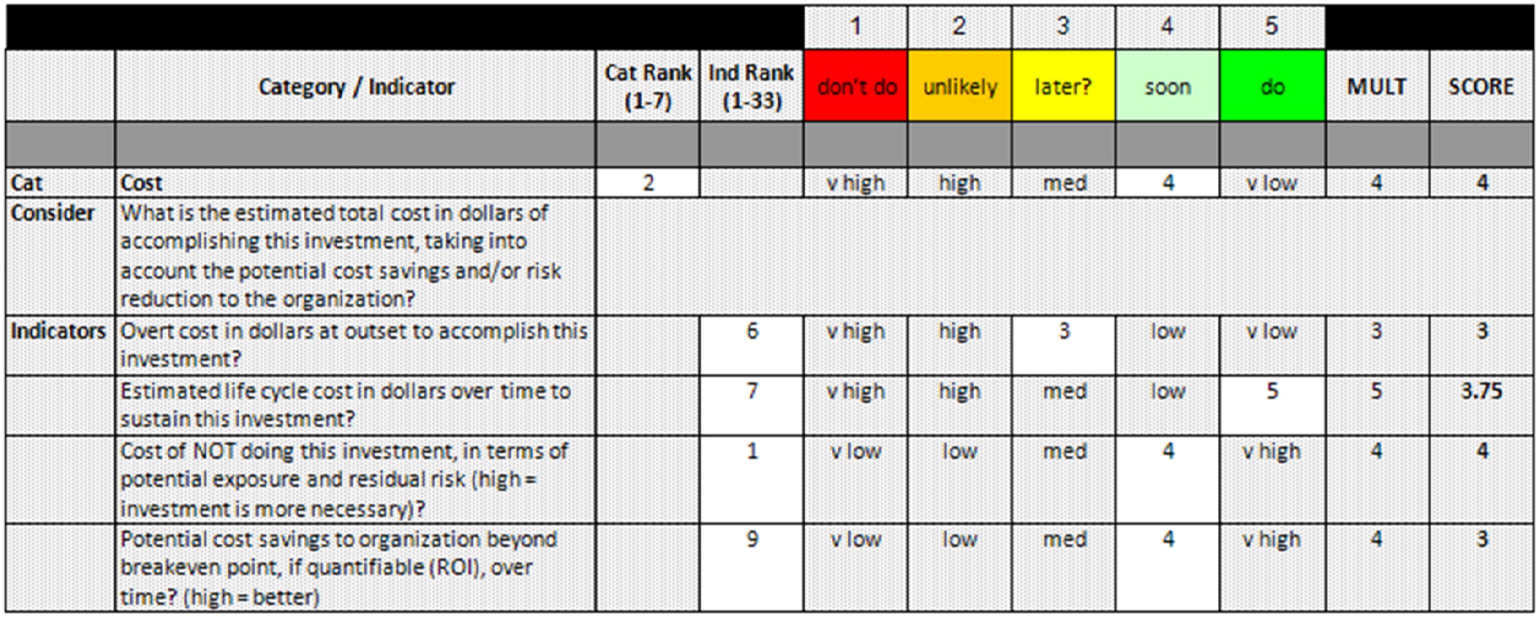
In this example and for the investment being considered, the answer to the Cost category question "What is the estimated total cost of accomplishing this investment . . ." is low. So the word "low" is replaced by the number "4" (indicated at the top of the column) to allow for a numeric calculation. Given the weighting factors that are applied based on the category rank, the score for the Cost category is calculated to be "4." This score is added to the other six category scores to arrive at the CAT TOTAL score that appears in Appendix C.3.

The indicators are assigned the following values and corresponding scores:

- Overt cost at outset is medium, so the word "med" is replaced by the number "3" and a resulting score of "3" is calculated based on the indicator rank.
- Estimated life-cycle cost is very low, so the word "v low" is replaced by the number "5" and a resulting score of "3.75" is calculated.
- Cost of NOT doing this investment is high, so the word "high" is replaced by the number "4" and a resulting score of "4" is calculated.
- Potential cost savings is high, so the word "high" is replaced by the number "4" and resulting score of "3" is calculated.

These indicator scores are added to the other 29 indicator scores for this investment to produce the IND TOTAL score that appears in Appendix C.3.

The red, orange, yellow, light green, and green colors and text are intended to serve as visual cues. Questions that have category and indicator answers that tend to the red end of the spectrum will likely result in a "don't do" this investment decision. Questions that have category and indicator answers that tend to the green end of the spectrum will likely result in a "do" this investment decision.



				1	2	3	4	5		
	Category / Indicator	Cat Rank (1-7)	Ind Rank (1-33)	don't do	unlikely	later?	soon	do	MULT	SCORE
Cat	Cost	2		v high	high	med	4	v low	4	4
Consider	What is the estimated total cost in dollars of accomplishing this investment, taking into account the potential cost savings and/or risk reduction to the organization?									
Indicators	Overt cost in dollars at outset to accomplish this investment?		6	v high	high	3	low	v low	3	3
	Estimated life cycle cost in dollars over time to sustain this investment?		7	v high	high	med	low	5	5	3.75
	Cost of NOT doing this investment, in terms of potential exposure and residual risk (high = investment is more necessary)?		1	v low	low	med	4	v high	4	4
	Potential cost savings to organization beyond breakeven point, if quantifiable (ROI), over time? (high = better)		9	v low	low	med	4	v high	4	3

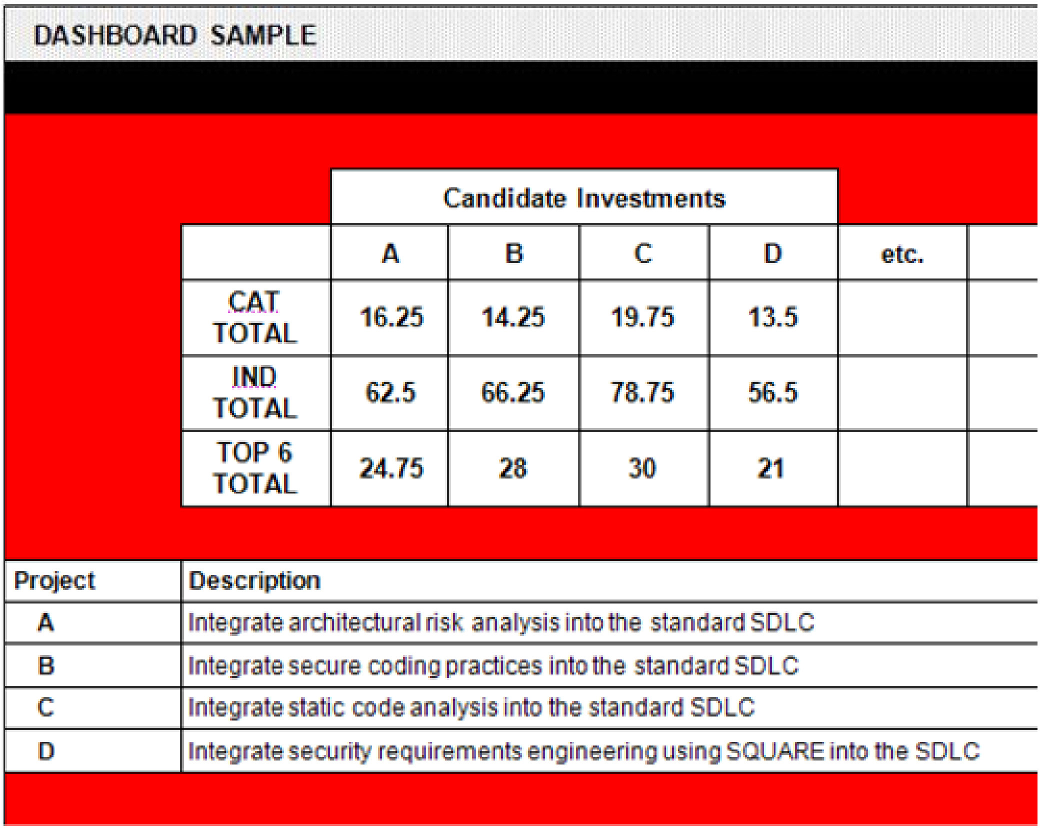
C.3 Summary Results for Four Investments

Summary results are calculated as follows:

- CAT TOTAL: the numeric sum of the scores for all 7 categories
- IND TOTAL: the numeric sum of the scores for all 33 indicators
- TOP 6 TOTAL: the numeric sum of the scores for the 6 highest priority indicators. This provides an alternative view in the event that 6 specific indicators are of equal or greater relevance to the investment decision.

The “Overall Summary View” provides a bar chart comparison of CAT TOTAL, IND TOTAL, and TOP 6 TOTAL. The elements of the Summary View are then displayed individually in the following Summary displays.

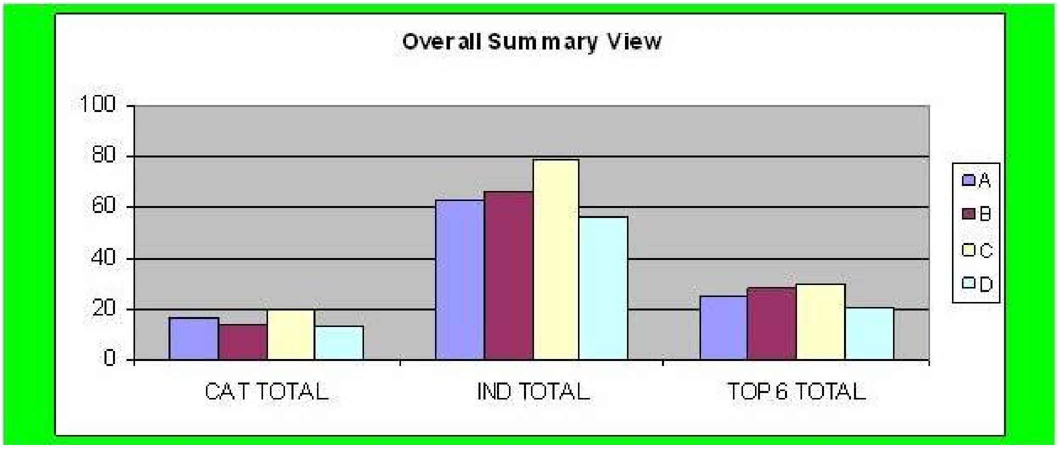
In this particular example, Investment C: *Integrate static code analysis into the standard SDLC* has the highest score (sum of CAT TOTAL and IND TOTAL; confirmed by TOP 6 TOTAL), so it should be considered as the first software assurance investment to fund. It is closely followed by Investment B: *Integrate secure coding practices into the standard SDLC*, which should be funded next assuming funds are available. Investment A: *Integrate architectural risk analysis into the standard SDLC* and Investment D: *Integrate security requirements engineering using SQUARE into the SDLC* are next in line respectively, subject to available resources.

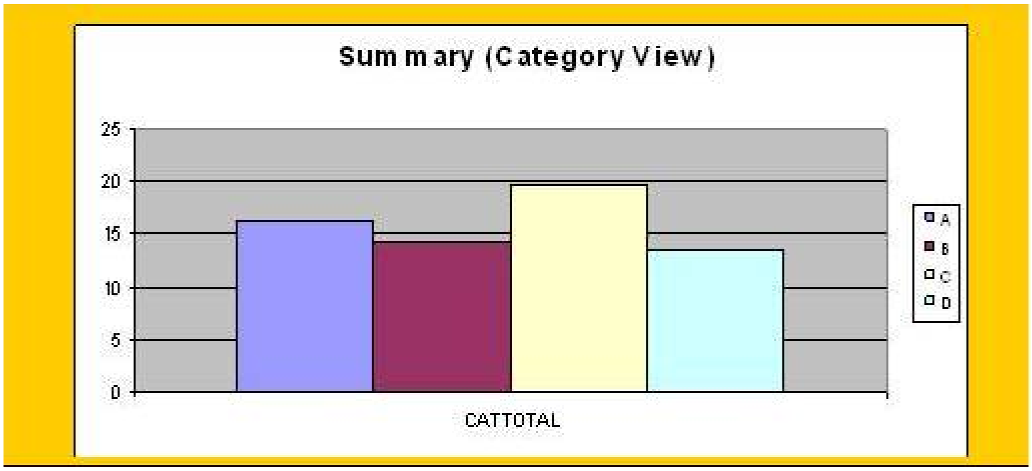


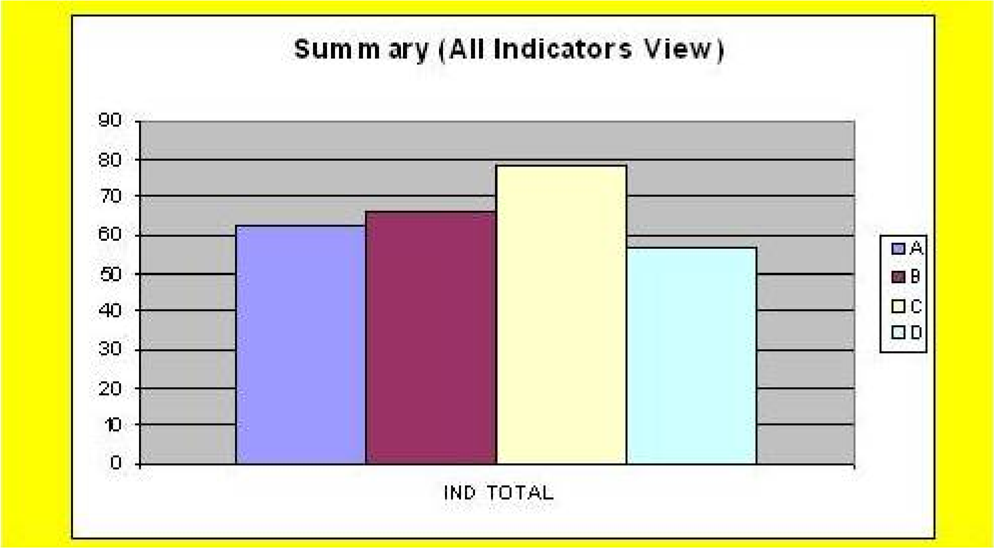
DASHBOARD SAMPLE

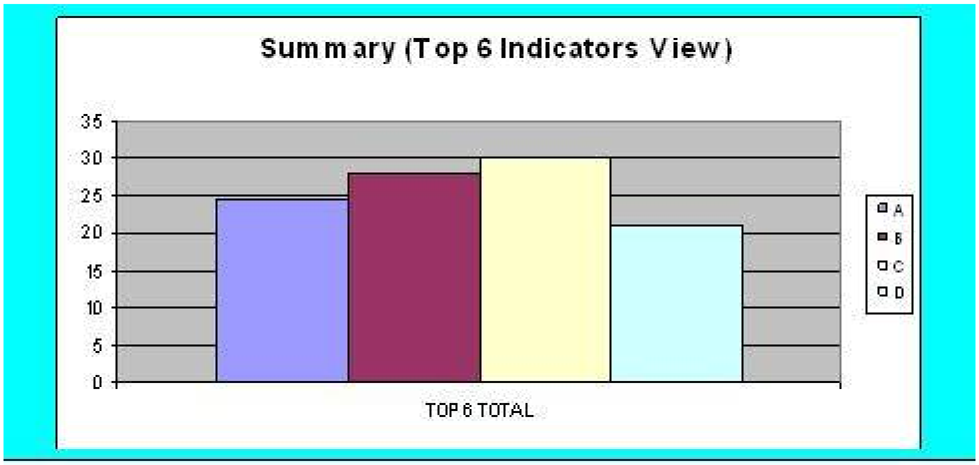
	Candidate Investments					
	A	B	C	D	etc.	
CAT TOTAL	16.25	14.25	19.75	13.5		
IND TOTAL	62.5	66.25	78.75	56.5		
TOP 6 TOTAL	24.75	28	30	21		

Project	Description
A	Integrate architectural risk analysis into the standard SDLC
B	Integrate secure coding practices into the standard SDLC
C	Integrate static code analysis into the standard SDLC
D	Integrate security requirements engineering using SQUARE into the SDLC


Overall Summary View
100
80
60
40
20
0
CAT TOTAL
IND TOTAL
TOP 6 TOTAL
A
B
C
D


Summary (Category View)
25
20
15
10
5
0
CATTOTAL
A
B
C
D


Summary (All Indicators View)
90
80
70
60
50
40
30
20
10
0
IND TOTAL
A
B
C
D


Summary (Top 6 Indicators View)
35
30
25
20
15
10
5
0
TOP 6 TOTAL
A
B
C
D

Appendix D: Process Improvement Background

A Short History of Process Improvement in Software

The software industry has been officially publishing process improvement models since 1987. At that time, the International Organization for Standards (ISO) published ISO 9000-13 for software. Then, in the same year, Watts Humphrey at the Software Engineering Institute (SEI) published an article about assessing software engineering capability [Humphrey 1987a], which would be developed into the earliest version of the Capability Maturity Model. That was described in *Characterizing the Software Process* [Humphrey 1987b] and *Managing the Software Process* [Humphrey 1989]. Subsequently version 1.0 of the CMM was released in August of 1991 through two technical reports, *Capability Maturity Model for Software* [Paulk 1991] and *Key Practices of the Capability Maturity Model* [Weber 1991]. This first edition would quickly become the generally accepted version of the Capability Maturity Model (SW-CMM v1.1, also known as the Software CMM), which was rolled out in 1993 [Paulk 1993].

Then in 1997, the SEI embarked on a new project to upgrade the application and expand the scope of the original CMM. The outcome was the Capability Maturity Model Integration for Systems Engineering/Software Engineering/Integrated Product and Process Development v1.1, which was simply called CMMI v1.1. This model is significantly different in form and application, and so it is in essence a different approach to process improvement. The initial version 1.0 was launched in 2000. However, the details of its implementation were laid out in a document entitled “Concept of Operations for the CMMI” [SEI 2001]. The CMMI SE/SW/IPPD/A (1.1) version was then published in 2002. Over the past seven years, several refinements have been published, the most recent version of which is the CMMI Product Suite Version 1.2 [SEI 2009b].

Two other process improvement models, which also embody a staged approach to the improvement of individual and small team capabilities, have to be mentioned here. Those are the Personal Software Process (PSP) [Humphrey 1997] and the Team Software Process (TSP) [Humphrey 2000]. These were both derived from the same general concept and principles as the CMM, and the PSP was actually developed in the same timeframe as CMM v1.1, i.e., 1990-1993. The TSP came later (1998). However, as the names suggest, these two approaches are focused on the improvement of individual and team capability rather than the overall organization. There is some indication from studies done by Capers Jones [Jones 2005] that these two approaches might be the most effective in terms of overall productivity payoffs. That fact might also imply that they would have commensurate payoffs in improving secure software performance. However, this has yet to be fully documented.

On the international front, throughout the 1990s the International Standards Organization (ISO) was working to develop a much more powerful model for software process improvement than its original quality standards, such as ISO 9000 and TickIT. This model was also originally intended to serve as a certification for software, in the same fashion as ISO 9000 did for manufacturing. That approach went under the informal name of SPICE throughout the 1990s. SPICE was formalized as the ISO 15504 Standard in May of 1998 (all except Part 5). However, over the past ten years, ISO 15504 has evolved into an assessment framework rather than an actual process improvement model. Therefore, its role is unclear. Nevertheless, it is still a very robust mechanism for deploying and evaluating practices in an organization’s software process.

Finally, there is ISO/IEC 21827:2008 [ISO 2008b]. ISO/IEC 21827 is specifically focused on installing and evaluating the maturity of the base practices needed to ensure security in a system or series of related systems. Since the model provides a complete set of life-cycle activities for developing a secure product, it cannot be ignored in a discussion such as this. In conjunction with the ISO standard, there is also a new and highly speculative industry model called the Building Security in Maturity Model [Chess 2009]. This model bills itself as providing the specific benchmarks for “developing and growing an enterprise-wide software security program.” It provides a structured set of practices that were derived from an analysis of the secure software processes of interviewed corporations. This model should be noted for the future.

Adopting a Staged Approach to Process Improvement

The concept that is most commonly is used to improve the capability of an organization's software process is the staged approach. It is normally based on progress through five distinct process maturity levels. With the exception of level 1, each of the maturity levels is characterized by a distinctive set of process areas (PAs). The degree of institutionalization of each of these process areas can be determined through assessment of five organizational factors called common features. The common features aspect of the staged approach measures an organization’s commitment to perform the specific set of key practices that have been defined for a given level of capability.

The common features aspect of the staged approach can be seen as establishing the basis for the proof that the organization is meeting the goals it defined for itself within each process area. Obviously, one of those goals could be a more secure product. The most visible concept in the staged approach is *the notion of maturity* levels, so we will address that first. The staged approach is concerned with building the most capable organization possible. It specifies five levels of increasing capability, from ad hoc, immature operation to mature, disciplined processes. And it is detailed in its specification of the technical requirements for this process. Thus, the staged approach provides an explicit upgrade path, which any organization can follow to reach a higher level of process maturity.

From a process improvement perspective, the working assumption that underpins the staged approach is that correct industry best practice is implemented by a common set of activities and assessed based on their documented presence or absence within the software process. That assessment can then be used to characterize the level of that organization’s capability. Given those presumptions, the graduated capability phases described in this model provide considerable benefit to the entire cast of characters in the software community. It offers any potential acquiring organization a standardized yardstick to measure a potential developer’s capabilities, and it gives the developing organization a mechanism to assess the capability of each potential subcontractor.

The staged approach to process improvement is grounded in a set of discrete key processes that can be used by any software organization to plan and manage their software operations. Thus, a number of people believe that the adoption of these particular processes can enhance any software organization’s ability to achieve and maintain optimal cost efficiency, as well as increase the quality of its products and the practical effectiveness of their processes.

Maturity Levels in a Staged Process to Support Security

It has yet to be established whether the increased efficiency obtained by reaching a particular capability level will result in better security. Nonetheless, it ought to be intuitively obvious that an organization with a well-defined and mature software process is more likely to produce correct code than one that is still functioning at the chaos level. The idea of maturity levels is easy enough to understand. As the organization refines the way it does business, it progresses through various levels of increasing capability. Each of these levels represents an explicit self-contained stratum of discrete practices that are installed in order to ensure continuous progress toward an ideal process.

Each of the process areas that are specified for a given level of maturity characterizes an additional function that is added to the process. That additional function adds one more additional capability, the eventual goal being to add enough capabilities to create a comprehensively effective software process. This is done in a staged fashion, in order to accommodate the practical realities of organizational development. Practices are added at each stage in a logical fashion. Foundational practices are added first and then these are followed at each stage by practices that continue to refine the software process. Likewise, the attainment of each individual level in the maturity model by definition means that the overall software process has moved to a higher level of proficiency. Those proficiency levels can also be used to judge whether that an organization’s software process is more likely to function properly and therefore produce more secure products.

Capability Clusters: Process Areas (PAs) and Their Support for Security

Each of the process areas in the staged approach designates “a cluster of related activities that, when performed collectively, achieve a set of goals considered important for enhancing process capability” [Paulk 1993]. Logically, a process area resides at just one explicit level of maturity. As such, each PA can be viewed as a particular capability that the organization must be able to document in order to demonstrate a given level of defined maturity. In essence, rather than dictating practice, PAs mark the characteristics of the software process that reside at a given maturity level. PAs are the logical place to operationalize secure software assurance practice. Many of the PAs from the CMMI are also important elements of the security landscape and so, for an organization that has reached an assessed level of maturity, it should be a simple matter of better focusing the traditional practices in that PA to establish more secure practice.

Appendix E: Improving Individual and Organizational Performance

Performance improvement measures the output of a process or procedure that could be modified to increase output, efficiency, or effectiveness. It is applied to individual performance or organizational performance in any business or organizational environment.

In order to bring about organizational change, a managed program of organizational development and performance improvement must be fostered by decision makers or a governance body of the organization. The program should include these activities:

1. Set goals for performance that will enable the organization to deliver software products and services that meet the utmost standards for software assurance.
2. Analyze organizational vision, mission, values, goals, and strategies. They can differ in the words used to communicate their strategic plan where individuals vary on how they define vision and values. However, words are less important than actions. It is important that the action is defined by the activities to reach the end state and that decision makers and employees agree on the definitions.
3. Assess the current environment. There are many effective tools and evaluation methods.
    a. Interviews are a popular tool to gather actual performance, feelings, causes, and solutions. If information is technical and detailed and emotionally charged, the interview can be conducted in person to establish a rapport and gather follow-up questions. It can be structured with forced choice or open-ended and probing questions.

    b. Observations can used to determine what is going on in the workplace. Observation is a powerful tool in capturing information on current skills, knowledge, and the context of the environment.

    c. Surveys can be used to solicit the thoughts, perceptions, and concerns of large groups of people. Surveys are anonymous and can generate honest results.

    d. Group processes such as focus groups use structured meetings as a means to obtain information and generate relationships among people from diverse areas. However, group processes can be difficult to lead and are not always successful. There are many types of group processes, such as force field analysis, and Delphi techniques.
4. Identify causes or factors that limit performance. What factors drive or cause performance? The cause of the lack of performance can be the lack of skill or knowledge in the field, lack of motivation among the employees at any level where they don't see how it is a good change, flawed or no incentives where professionals are burdened with excessive work, automatic salary increases, no pay performance, managers ignoring or not valuing efforts of individuals, or a flawed ergonomic setting.
5. Design and develop solutions to address the causes. Lack of skill or knowledge can be addressed by training, well-written documentation, coaching and mentoring, and knowledge management or decision tools that support activities needed to meet responsibilities. Address motivation by providing information for employees to recognize the benefits, impact, and value of change. Use role models and early successes to support confidence and empower

individual participation in selecting goals. Effective incentives may require revised policies and contract relationships, in addition to management training and incentives, recognition, and bonus programs. Environmental solutions must address ergonomic improvements and incorporate work process and redesign, the latest tools and technologies, and greater emphasis on job-person matches.

6. Implement the solutions for better performance. Interventions for performance support involve instructional methods for learning the organization, action learning, and self-directed learning and training, continuing education through distance learning and non-instructional support uses of job aids, electronic support systems, job specifications and standards. Furthermore, job analyses and work design should be addressed by job specifications, job rotation, work methods, continuous improvement, ergonomics, preventative maintenance, and safety.

   Personal development can involve mentoring and coaching, career development and assessment, and feedback. A human resource development program should have in place the recommended processes for selection and staffing, financial planning, health and wellness, motivation for incentives and rewards, performance appraisals, assessment centers and competency testing, succession planning and career pathing, leadership and executive development, and management and supervisor development.

7. Evaluate the interventions to verify whether they are strengthening software assurance workplace performance. Formative evaluations help shape or mold ongoing processes to provide information for improvement. It is iterative and ongoing to encourage continuous improvement in procedures and methods. A summative evaluation focuses on the effectiveness of a performance intervention once it is implemented. It measures reaction and personal acquisition of knowledge and skills. Summative evaluation should take place during implementation and change.

Appendix F: Relevance of Social Science to the Business Case

When we think about business case, we often think in terms of hard economic data. However, there are social science findings which are also relevant to business case. An understanding of these findings can help the reader who is trying to gain organizational support for developing the business case for software assurance.

Consultants are taught valuable social science related skills to better understand their clients' true requirements. One example is the "Goal Hierarchy of Needs," which should not be confused with Maslow's "Hierarchy of Needs." The concept is very simple, with the rationale that whatever decision people make in the business world, it is made in a hierarchy of needs, with the most important need or goal being an emotion or emotions. Clients must understand these emotions if they are to really meet the requirements in all ways. For example, if someone has a need for a new online transaction system, then their higher need could be because they want to make more sales, with the higher need to make more money that they can use to invest, pay dividends, etc. The decision maker, however, is a person with emotional drivers, which in this case may be to feel more secure or obtain more power. In the case of investment in security or software assurance, you could argue it is an easier sell to someone who wants to feel more secure than someone who wants to feel more powerful.

Investment in security is seen by many as an insurance policy to mitigate risk rather than simply a professional thing to do. The former is the rational part of our brain, whereas the latter is the emotional component of pride and ethics. Both have a part to play in any business decision, as demonstrated in the following design example. A famous designer once said that good design does not have to imply more cost or expense. He described good design as having two aspects: to satisfy people's needs and people's wants. The "needs" relate to the functional aspects of design. Does it do what I want it to do? Does it play MP3 music? is an example. The "want" relates to the emotional aspects of design. Does it look good? Does it make me feel good? There are many emotions related to feeling good, such as pride and achievement. Apple is recognized as understanding this aspect very well, which is why their MP3 players are now being emulated by many others. One could argue that feeling secure is also an aspect of feeling good, which in turn satisfies the "wants" of some people. These examples show that understanding emotions is important in any business decision and argue that motivators and demotivators have an important emotional content.

There are many other examples that could be used to help make and position the business case for security and software assurance, but this is not the place to describe them. However, this area is starting to attract some serious research and comment. This statement by Bruce Schneier makes the point well [Schneier 2008a]: "Security is both a feeling and a reality. And they're not the same."

Bibliography

[Alberts 2002]

Alberts, Christopher & Dorofee, Audrey. *Managing Information Security Risks: The OCTAVE ® Approach*. Addison Wesley, 2002. http://www.cert.org/octave.

[Allen 2005]

Allen, Julia. *Governing for Enterprise Security* (CMU/SEI-2005-TN-023). Software Engineering Institute, Carnegie Mellon University, June 2005.
http://www.sei.cmu.edu/publications/documents/05.reports/05tn023.html

[Allen 2008a]

Allen, Julia. "Making Business-Based Security Investment Decisions – A Dashboard Approach." Build Security In, 2008. https://buildsecurityin.us-cert.gov/daisy/bsi/articles/best-practices/management/985-BSI.html

[Allen 2008b]

Allen, Julia H., Barnum, Sean, Ellison, Robert, McGraw, Gary, & Mead, Nancy R. *Software Security Engineering: A Guide for Project Managers*. Addison-Wesley, 2008 (ISBN 978-0-321-50917-8).

[Anderson 2001]

Anderson, Ross. *Why Information Security Is Hard–An Economic Perspective.* Computer Laboratory, University of Cambridge, 2001.

[Apfel 2003]

Apfel, Audrey. "The Total Value of Opportunity Approach." *CIO*, January 15, 2003.

[Arora 2008]

Arora, Ashish, Frank, Steven, & Telang, Rahul. "Estimating Benefits from Investing in Secure Software Development." Build Security In, 2008. https://buildsecurityin.us-cert.gov/daisy/bsi/articles/knowledge/business/267-BSI.html

[Bailey 2003]

Bailey, John & Heidt, Stephen R. "Why Is Total Cost of Ownership (TCO) Important?" *CSO*, November 2003.

[Bailey 2008a]

Bailey, John, Drommi, Antonio, Ingalsbe, Jeffrey, Mead, Nancy R., & Shoemaker, Dan. "Models for Assessing the Cost and Value of Software Assurance." Build Security In, February 2007.
https://buildsecurityin.us-cert.gov/daisy/bsi/articles/knowledge/business/684-BSI.html

[Bailey 2008b]

Bailey, John, Drommi, Antonio, Ingalsbe, Jeffrey, Mead, Nancy, & Shoemaker, Dan. "A Common Sense Way to Make the Business Case for Software Assurance." Build Security In, 2008.
https://buildsecurityin.us-cert.gov/daisy/bsi/articles/knowledge/business/952-BSI.html

[Bartol 2008]
Bartol, Nadia. *Practical Measurement Framework for Software Assurance and Information Security*, DHS Software Assurance Forum Measurement Working Group, Draft Report, October 2008. https://buildsecurityin.us-cert.gov/swa/downloads/SwA_Measurement.pdf

[Benson 1992]
Benson, Robert J. *Information Economics and the Business Value of Computers* (POSPP Report P-34-1). Dallas, TX: Chantico Publishing, January 1992.

[Berinato 2002]
Berinato, S. “Finally, a Real Return on Security Spending.” *CIO Magazine* (Australia), August 4, 2002.

[Berkman 2002]
Berkman, Eric. “How to Use the Balanced Scorecard.” *CIO*, May 15, 2002.

[Bitterman 2006]
Bitterman, Michael. *How IT Benefits From Adopting Measurement-Management Techniques*. ITPMG, 2006.

[Blum 2006]
Blum, D. *Making Business Sense of Information Security, Security and Risk Management Strategies*. Burton Group, Version 1.0, February 10, 2006.

[Boehm 2001]
Boehm, Barry & Basili, Victor R. “Software Defect Reduction Top 10 List.” *IEEE Computer, vol. 34*, 1 (January 2001): 135–137. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=00962984

[Borland 2005]
Borland Software Corporation. “Software Delivery Optimization: Maximizing the Business Value of Software,” Borland Vision and Strategy Solution Whitepaper, 2005.

[Brynjolfsson 2003]
Brynjolfsson, Erik & Hitt, Lorin M. *Computing Productivity: Firm-Level Evidence* (Working Paper No 4210-01). Cambridge, MA: MIT Sloan School of Management, June 2003.

[Campbell 2006]
Campbell, George K. “Measures and Metrics in Corporate Security: Communicating Business Value.” CSO Executive Council, 2006.
https://www.csoexecutivecouncil.com/content/Metrics_Mini_Update_060706.pdf

[Cavusoglu 2006]
Cavusoglu, Huseyin; Cavusoglu, Hasan; & Zhang, Jun. “Economics of Security Patch Management.” Workshop on Economics of Information Security (WEIS). Cambridge, England: Robinson College, University of Cambridge, June 2006.

[Chess 2009]
Chess, Brian, McGraw, Gary, & Migues, Sammy. Building Security In Maturity Model. Creative Commons, 2009. http://www.bsi-mm.com/

[CISWG 2005]
Corporate Information Security Working Group. Adam H. Putnam, Chairman; Subcommittee on Technology, Information Policy, Intergovernmental Relations & the Census Government Reform Committee, U.S. House of Representatives. “Report of the Best Practices and Metrics Teams.” November 17, 2004; updated January 10, 2005.
http://www.educause.edu/ir/library/pdf/CSD3661.pdf

[CMS 2002]
Centers for Medicare & Medicaid Services (CMS). “CMS Information Security Risk Assessment (RA) Methodology, Version1.1.” CMS Office of Information Services (OIS), Security and Standards Group (SSG), 2002. http://csrc.nist.gov/groups/SMA/fasp/documents/risk_mgmt/RA_meth.pdf

[CNSS 2006]
Committee on National Security Systems. National Information Assurance (IA) Glossary, Instruction No. 4009. Ft. Meade, MD: CNSS Secretariat, June 2006.
http://www.cnss.gov/Assets/pdf/cnssi_4009.pdf

[Colbert 2002]
Colbert, Edward, Reifer, Don, & Gangadharan, Murali. “COCOMO II Security Extensions.” 17th International Forum on COCOMO and Software Cost Modeling. Los Angeles, CA, October 2002.

[Colbert 2006]
Colbert, Edward & Wu, Danni. “Costing Secure Systems Workshop Report.” 21st International Forum on COCOMO & Software Cost Modeling, November 2006.
http://sunset.usc.edu/events/2006/CIIForum/pages/presentations
/outbrief - Costing-Secure-Systems-Workshop-Report.2006 _Nov_07.ppt

[Construx 1998]
Construx Software Builders website, www.construx.com, 1998, cited in Shoemaker, D. and Vladan Jovanovic, “Engineering a Better Software Organization,” Quest Publishing House, Ann Arbor, 1998.

[DiCarlo 2003]
DiCarlo, Lisa. “Best Countries for Outsourcing.” Forbes.com, August 27, 2003.
http://www.forbes.com/2003/08/27/cx_ld_0827bestcountries.html

[Dolan 2006]
Dolan, Kerry A. “Offshoring the Offshorers.” Forbes.com, April 17, 2006.
http://www.forbes.com/free_forbes/2006/0417/074.html

[Drugescu 2006]
Drugescu, Cezar. “Maximizing the Return on Investment of Information Security Programs: Program Governance and Metrics.” Information Systems Security Journal, Taylor & Francis, December 2006.

[Dynes 2006]
Dynes, Scott; Andrijcic, Eva; & Johnson, M. Eric. “Cost to the U.S. Economy for Information Infrastructure Failures.” Workshop on Economics of Information Security (WEIS). Cambridge, England: Robinson College, University of Cambridge, June 2006.

[Eisenberg 2003]
Eisenberg, Bryan. “How Are You Measuring Customers?” *ROI Marketing*, February 2003.

[Feather 2001]
Feather, M. S.; Sigal, B.; Cornford, S. L.; & Hutchinson, P. "Incorporating Cost-Benefit Analyses into Software Assurance Planning," 62-68. *Proceedings of the 26th Annual NASA Goddard Software Engineering Workshop*, IEEE Computer Society, 2001.

[Goertzel 2007]
Goertzel, Karen Mercedes, Winograd, Theodore, McKinley, Holly Lynne, Oh, Lyndon, Colon, Michael, McGibbon, Thomas, Fedchak, Elaine, & Vienneau, Robert. *Software Security Assurance: A State-of-the-Art Report (SOAR)*. Information Assurance Technology Analysis Center (IATAC) and Defense Technical Information Center (DTIC), 2007.
http://iac.dtic.mil/iatac/download/security.pdf

[Gordon 2006]
Gordon, L. A. & Loeb, M. P. “Budgeting Process for Information Security Expenditures.” *Communications of the ACM 49*, 1 (January 2006): 121-125.

[Higaki 2008]
Higaki, Wes & Howard, Michael. “An Overview of SAFECode and Best Practices for Secure Development.” DHS Software Assurance Forum. Gaithersburg, Md., October 14–16, 2008.
https://buildsecurityin.us-cert.gov/swa/downloads/Howard.pdf

[Hofstede 1997]
Hofstede, G. *Cultures and Organizations: Software of the Mind*. McGraw-Hill, 1997.

[Howard 1998]
Howard, John & Longstaff, Tom. *A Common Language for Computer Security Incidents*. Sandia National Laboratories Report SAND98-8667, October 1998.
http://www.cert.org/research/taxonomy_988667.pdf

[Howard 2003]
Howard, Michael & LeBlanc, David. *Writing Secure Code*, 2nd edition. Microsoft Press, 2003.

[Howard 2006]
Howard, Michael & Lipner, Steve. *The Security Development Lifecycle – SDL: A Process for Developing Demonstrably More Secure Software*. Microsoft Press, 2006.

[Huang 2006]
Huang, C. D., Hu, Q., & Behara, R. S. “Economics of Information Security Investment in the Case of Simultaneous Attacks.” The Fifth Workshop on the Economics of Information Security (WEIS 2006). University of Cambridge, England, June 26-28, 2006.

[Hubbard 1997]
Hubbard, Douglas. “Everything is Measurable.” *CIO Enterprise Magazine,* November 1997.

[Hubbard 1999]
Hubbard, Douglas. “Checks and Balances: Measuring the Value of Technology Investment.” *CIO,* April 15, 1999.

[Humphrey 1987a]
Humphrey, Watts S. *A Method for Assessing the Software Engineering Capability of Contractors* (CMU/SEI-87-TR-023). Software Engineering Institute, Carnegie Mellon University, 1987. http://www.sei.cmu.edu/publications/documents/87.reports/87.tr.023.html

[Humphrey 1987b]
Humphrey, Watts S. *Characterizing the Software Process: A Maturity Framework* (CMU/SEI-87-TR-11). Software Engineering Institute, Carnegie Mellon University, June 1987.

[Humphrey 1989]
Humphrey, Watts S. *Managing the Software Process*, Addison-Wesley, 1989.

[Humphrey 1997]
Humphrey, Watts S. *Introduction to the Personal Software Process*. Reading, MA: Addison-Wesley Publishers, 1997 (ISBN: 0201548097)

[Humphrey 2000]
Humphrey, Watts S. *Introduction to the Team Software Process*. Reading, MA: Addison-Wesley Publishers, 2000 (ISBN: 020147719X)

[Ingalsbe 2008]
Ingalsbe, Jeffrey A., Shoemaker, Dan, Mead, Nancy R., & Drommi, Antonio. “Threat Modeling the Enterprise.” *AMCIS 2008 Proceedings*, Paper 133, 2008. http://aisel.aisnet.org/amcis2008/133

[ISO 1998a]
International Organization for Standardization. *Information Technology — System and Software Integrity Levels*, ISO/IEC 15026, 1998.

[ISO 1998b]
International Organization for Standardization. *Software Process Improvement and Capability dEtermination – SPICE*, ISO/IEC 15504, 1998.

[ISO 2007]
International Organization for Standardization. “ISO/IEC 27001 & 27002: Implementation Guidance and Metrics, Version 0.7.” ISO 27001 Security Implementers’ Forum, June 5, 2007.

[ISO 2008a]
International Organization for Standardization. *Information technology – Security techniques – Information security risk management*, ISO/IEC 27005, First edition, June 15, 2008.

[ISO 2008a]
International Organization for Standardization. *Information technology – Security techniques – Systems Security Engineering — Capability Maturity Model® (SSE-CMM®)*, ISO/IEC 21827:2008, Edition 2, 2008.

[Jaquith 2002]
Jaquith, Andrew. *The Security of Applications: Not All Are Created Equal* @atstake Security Research Report, 2002.

[Jones 1994]
Jones, Capers. *Assessment and Control of Software Risks*. Prentice-Hall: Englewood Cliffs, NJ, 1994.

[Jones 2005a]
Jones, Capers. "Software Quality in 2005, a Survey of the State of the Art." Software Productivity Research, Marlborough, Massachusetts, 2005.

[Jones 2005b]
Jones, Jack. "An Introduction to Factor Analysis of Information Risk (FAIR): A Framework for Understanding, Analyzing, and Measuring Information Risk," 2005.
http://riskmanagementinsight.com/media/documents/FAIR_Introduction.pdf

[Kaplan 1992]
Kaplan, Robert S. & Norton, David P. "The Balanced Scorecard: Measures That Drive Performance." *Harvard Business Review 70*, 1 (Jan.-Feb. 1992).

[Kaplan 1993]
Kaplan, Robert & Norton, David. "Putting the Balanced Scorecard to Work" *Harvard Business Review,* September-October 1993: 134-147.

[Kaplan 1996]
Kaplan, Robert & Norton, David. "Using the Balanced Scorecard as a Strategic Management System." *Harvard Business Review* (Jan-Feb 1996): 75-85.

[Karabacak 2005]
Karabacak, Bilge & Sogukpinar, Ibrahim. "ISRAM: Information Security Risk Analysis Method." *Computers & Security 24,* 2 (March 2005).

[Kitchenham 1996]
Kitchenham, Barbara & Pfleeger, Shari Lawrence. "Software Quality, the Elusive Target", *Software 13*, 1 (January 1996): 12-21.

[Kleinfeld 2006]

Kleinfeld, Abe. “Measuring Security.” *Information Systems Security Journal*, Taylor & Francis, November 2006.

[KPMG 1996]

KPMG Technology and Services Group website, www.kpmg.ca 1996, cited in Shoemaker, D. and Vladan Jovanovic, “Engineering a Better Software Organization,” Quest Publishing House, Ann Arbor, 1998.

[Konary 2005]

Konary, Amy. “Atos Origin: A Microsoft Software Assurance Case Study.” Framingham: MA: International Data Group (IDG), August 2005.

[Kwon 2001]

Kwon, Regina. “The Probability Problem.” *Baseline Magazine,* December 2001.

[Lacity 1993]

Lacity, Mary C. & Hirschheim, Rudy. *Information Systems Outsourcing*. John Wiley & Sons, 1993 (ISBN-13: 978-0471938828).

[Leahy 2002]

Leahy, Tad. “IT Measures Evolve.” *Business Finance,* March 2002.

[Lipner 2005]

Lipner, Steve & Howard, Michael. *The Trustworthy Computing Security Development Lifecycle*. Microsoft Developer Network, March 2005. http://msdn.microsoft.com/en-us/library/ms995349.aspx

[Luehrman 1998a]

Luehrman, Timothy A. “Investment Opportunities as Real Options: Getting Started on the Numbers.” *Harvard Business Review,* July-August 1998.

[Luehrman 1998b]

Luehrman, Timothy A., “Strategy as a Portfolio of Real Options.” *Harvard Business Review,* September-October 1998.

[Madachy 2002]

Madachy, Ray and Stutzke, Dick. “Use of Cost Models in Risk Management.” 17th International Forum on COCOMO and Software Cost Modeling. Los Angeles, CA, Oct. 2002.

[Mahmood 2004]

Mahmood, Mo Adam; Kohli, Rajiv; & Devaraj, Sarv. “Measuring Business Value of Information Technology in E-Business Environments.” *Journal of Management Information Systems 21*, 1 (Summer 2004): 11-16.

[Martin ND]

Martin, James R. “The Balanced Scorecard Concepts.” Management and Accounting Web. http://www.maaw.info/BalScoreSum.htm

[Mayor 2002]

Mayor, Tracy. “A Buyer’s Guide to IT Value Methodologies.” *CIO Magazine*, July 2002.

[McClure 2003]

McClure, Ben. “All About EVA.” Investopedia.com, March 2003.

[McGibbon 1999]

McGibbon, Thomas. “A Business Case for Software Process Improvement Revised.” DoD Data Analysis Center for Software (DACS), 1999.

[McGraw 2005]

McGraw, Gary. “Risk Management Framework (RMF).” Build Security In, 2005. https://buildsecurityin.us-cert.gov/daisy/bsi/articles/best-practices/risk/250-BSI.html

[McGraw 2006]

McGraw, Gary. *Software Security: Building Security In*. Addison-Wesley Professional, 2006.

[Mead 2003]

Mead, Nancy R. *International Liability Issues for Software Quality* (CMU/SEI-2003-SR-001). Software Engineering Institute, Carnegie Mellon University, June 2003.

[Mead 2008]

Mead, Nancy R. “SQUARE Process.” Build Security In, 2008. https://buildsecurityin.us-cert.gov/daisy/bsi/articles/best-practices/requirements/232-BSI.html

[Meftah 2008]

Meftah, Barmak. “Business Software Assurance: Identifying and Reducing Software Risk in the Enterprise.” 9th Semi-Annual Software Assurance Forum, Gaithersburg, Md., October 2008. https://buildsecurityin.us-cert.gov/swa/downloads/Meftah.pdf

[Meta Group 2000]

Meta Group. “Leadership Beyond the Project.” Stamford, CT: Meta Group, Gartner, 2000.

[Microsoft 2005]

Microsoft. “Build an Airtight Business Case for New IT Investments.” Redmond, WA: Microsoft, December 2005.

[Microsoft 2009]

Microsoft Corp. *The Microsoft Security Development Lifecycle (SDL): Measurable Improvements for Flagship Microsoft Products*. http://msdn.microsoft.com/en-us/security/cc424866.aspx (2009).

[MITRE 2009]

MITRE. Common Weakness Enumeration. http://cwe.mitre.org/data/index.html (2009).

[Nagaratnam 2005]

Nagaratnam, N., Nadalin, A., Hondo, M., McIntosh, M., & Austel, P. "Business-Driven Application Security: From Modeling to Managing Secure Applications." *IBM Systems Journal 44*, 4 (2005): 847-867.

[Neely 2001]

Neely, James. "Hybrid Real Options Valuation of Risky Product Development Projects." Cleveland, OH: Booz-Allen & Hamilton, 2001.

[NSTISSC 1998]

National Security Telecommunications and Information Systems Secure Committee. *Index of National Security Telecommunications Information Systems Security Issuances* (NSTISSI No. 4014). Ft. Mead, MD: NSTISSC Secretariat, January 1998.

[O'Neill 2007]

O'Neill, Don. "Calculating Security Return on Investment." Build Security In, 2007.
https://buildsecurityin.us-cert.gov/daisy/bsi/articles/knowledge/business/677-BSI.html

[Ozment 2006]

Ozment, Andy & Schechter, Stuart E. "Economic Barriers to Adopting New Security Protocols." Workshop on Economics of Information Security (WEIS), Cambridge, England: Robinson College, University of Cambridge, June 2006.

[Park 2006]

Park, Alvin R. "Determining the Value of Microsoft Software Assurance." Gartner, April 2006.

[Parker 1989]

Parker, Marilyn M & Benson, Robert J. "Enterprisewide Information Economics." *Journal of Information Systems Management 6*, 4 (Fall 1989): 7-13.

[Paulk 1991]

Paulk, M., Curtis, B., Chrissis, M. B., et al. *Capability Maturity Model for Software* (CMU/SEI-91-TR-024). Software Engineering Institute, Carnegie-Mellon University, 1991.

[Paulk 1993]

Paulk, M., Curtis, B., Chrissis, M., & Weber, C. *Capability Maturity Model for Software, Version 1.1* (CMU/SEI-93-TR-024). Software Engineering Institute, Carnegie-Mellon University, 1993.

[Pettit 2001]

Pettit, Justin; Dower, John; Pichler, Karl; & Perez, Jorge. "EVA and Corporate Portfolio Strategy." *EVAluation 3*, 9 (December 2001).

[PITAC 1999]

President's Information Technology Advisory Committee (PITAC). "Information Technology Research: Investing in Our Future." Arlington, VA: President's Information Technology Advisory Committee, February 1999.

[Rauscher 2006]
Rauscher, Karl F., Krock, Richard E., & Runyon, James P. "Eight Ingredients of Communications Infrastructure: A Systematic and Comprehensive Framework for Enhancing Network Reliability and Security." *Bell Labs Technical Journal 11*, 3 (2006): 73–81.

[Reifer 2006]
Reifer, Donald. "CONIPMO Workshop." Practical Software Measurement Conference, Vail, CO, July 24-28, 2006.

[REF 2008]
CERT® Resiliency Engineering Framework, Preview version, v0.95R. Carnegie Mellon University, Software Engineering Institute, March 2008. http://www.cert.org/resiliency_engineering

[Ruefle 2008]
Ruefle, Robin. "Defining Computer Security Incident Response Teams." Build Security In, 2008. https://buildsecurityin.us-cert.gov/daisy/bsi/articles/best-practices/incident/662-BSI.html

[SAFECode 2008]
Simpson, Stacy, Editor. "Fundamental Practices for Secure Software Development: A Guide to the Most Effective Secure Development Practices in Use Today." SAFECode, October 2008. http://www.safecode.org/publications/SAFECode_Dev_Practices1108.pdf

[Schneier 2008a]
Schneier, Bruce. "The Psychology of Security." Schneier.com, January 18, 2008. http://www.schneier.com/essay-155.html

[Schneier 2008b]
Schneier, Bruce. "Security ROI." *Schneier on Security*, September 2, 2008. http://www.schneier.com/blog/archives/2008/09/security_roi_1.html

[SEI 2001]
Software Engineering Institute. "Concept of Operations for the CMMI," 2001. http://www.sei.cmu.edu/cmmi/background/conops.html

[SEI 2008]
Software Engineering Institute and CyLab. *Proceedings of the Making the Business Case for Software Assurance Workshop*. Pittsburgh, PA September 26, 2008. Software Engineering Institute and CyLab, 2008. https://buildsecurityin.us-cert.gov/daisy/bsi/1074-BSI.html

[SEI 2009a]
Software Engineering Institute. *CMMI Appraisals*, 2009. http://www.sei.cmu.edu/cmmi/appraisals/index.html

[SEI 2009b]
Software Engineering Institute. *CMMI Models and Reports*, 2009. http://www.sei.cmu.edu/cmmi/models/index.html

[Shao 2007]

Shao, Benjamin B. M. & David, Julie Smith. "The Impact of Offshore Outsourcing on IT Workers in Developed Countries." *Communications of the ACM 50*, 2 (February 2007): 89–94.

[Soo Hoo 2001]

Soo Hoo, K., Sudbury, A. W., & Jaquith, A. R. "Tangible ROI Through Secure Software Engineering." *Secure Business Quarterly 1*, 2 (Fourth Quarter 2001).

[SSE-CMM 2003]

Systems Security Engineering Capability Maturity Model (SSE-CMM), Version 3.0, 2003.

[Stolen 2002]

Stolen, K., den Braber, F., & Dimitrakos T. "Model-based Risk Assessment – The CORAS Approach," 2002. http://www.nik.no/2002/Stolen.pdf

[Stoneburner 2002]

Stoneburner, Gary, Goguen, Alice, & Feringa, Alexis. *Risk Management Guide for Information Technology Systems* (NIST Special Publication 800-30). National Institute of Standards and Technology, July 2002. http://csrc.nist.gov/publications/PubsSPs.html

[Suh 2003]

Suh, Bomil & Han, Ingoo. "The IS Risk Analysis Based on a Business Model." *Information & Management 41* (2003): 149-158. http://www.sirim.my/techinfo/P2/Text/img_p149.pdf

[Vorster 2005]

Vorster, Anita & Labuschagne, Les. "A Framework for Comparing Different Information Security Risk Analysis Methodologies," 95-103. *Proceedings of the 2005 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries*. ACM, 2005 (ISBN 1-59593-258-5).

[Wang 2006]

Wang, R. & Erickson, J. "The Financial Impact of Packaged Applications, a Tool for Comparing the ROI of Enterprise Applications." Cambridge, MA: Forrester Research, Inc, July 2006.

[Weber 1991]

Weber, C. V., Paulk, M. C., Wise, C. J., & Withey, J. V. *Key Practices of the Capability Maturity Model* (CMU/SEI-91-TR-25, ADA240604). Software Engineering Institute, Carnegie Mellon University, August 1991. http://www.sei.cmu.edu/publications/documents /91.reports/91.tr.025.html

[West 2004]

West, Richard & Daigle, Stephen L. "Total Cost of Ownership: A Strategic Tool for ERP Planning and Implementation." Philadelphia, PA: Center for Applied Research (CFAR), January 2004.

[Westby 2007]

Westby, Jody R. & Allen, Julia H. *Governing for Enterprise Security (GES) Implementation Guide* (CMU/SEI-2007-TN-020). Software Engineering Institute, Carnegie Mellon University, August 2007. http://www.sei.cmu.edu/publications/documents/07.reports/07tn020.html

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY
(Leave Blank)

2. REPORT DATE
April 2009

3. REPORT TYPE AND DATES COVERED
Final

4. TITLE AND SUBTITLE
Making the Business Case for Software Assurance

5. FUNDING NUMBERS
FA8721-05-C-0003

6. AUTHOR(S)
Nancy R. Mead, Julia H. Allen, W. Arthur Conklin, Antonio Drommi, John Harrison, Jeff Ingalsbe, James Rainey, Dan Shoemaker

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

8. PERFORMING ORGANIZATION REPORT NUMBER
CMU/SEI-2009-SR-001

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
HQ ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2116

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

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

12B DISTRIBUTION CODE

13. ABSTRACT (MAXIMUM 200 WORDS)

This report provides guidance for those who want to make the business case for building software assurance into software products during each software development life-cycle activity. The business case defends the value of making additional efforts to ensure that software has minimal security risks when it is released and shows that those efforts are most cost-effective when they are made appropriately *throughout* the development life cycle. Although there is no single model that can be recommended for making the cost/benefit argument, there are promising models and methods that can be used individually and collectively for this purpose, as well as some convincing case study data that supports the value of building software assurance into newly developed software. These are described in this report.

The report includes a discussion of the following topics as they relate to the business case for software assurance: cost/benefit models, measurement, risk, prioritization, process improvement, globalization, organizational development, and case studies. These topics were selected based on earlier studies and collaborative efforts, as well as the workshop "Making the Business Case for Software Assurance," which was held at Carnegie Mellon University in September 2008.

14. SUBJECT TERMS
software assurance, business case, return on investment, ROI, software security, cost/benefit

15. NUMBER OF PAGES
118

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
Unclassified

18. SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT
Unclassified

20. LIMITATION OF ABSTRACT
UL