If one wants to pass through open doors easily, one must bear in mind that they have a solid frame: this principle, according to which the old professor had always lived is simply a requirement of the sense of reality. But if there is such a thing as a sense of reality - and no one will doubt that it has its raison d'être - then there must be something that one can call a sense of possibility. Anyone possessing it does not say, for instance: Here this or that happened, will happen, must happen. He uses his imagination and says: Here such and such might, should or ought to happen. And if he is told that something is the way it is, then he thinks: Well, it could probably just as easily be some other way. So the sense of possibility might be defined outright as the capacity to think how everything could 'just as easily' be, and to attach no more importance to what is than to what is not.


Joe Dawson was increasingly becoming comfortable with information system security concepts. Indeed it had been an interesting adventure...to learn more about security. Matt Bishop’s book had introduced some very serious concepts in technical information system security. Joe was indeed thankful to Randy for having pointed him in...
that direction. However given that Joe was not really a technical person, Bishop’s book was not an easy read. He still had a lot of unanswered questions. In particular Joe was unsure how security could be built into new software development. This was becoming an issue for Joe since SureSteel was in the process of evaluating vendors for custom developing backend systems.

Joe had often read in the popular press and elsewhere that most software development considered security as an afterthought. What did that mean? What was the right way? Joe was not even sure about the notion of controls. From a commonsense perspective he did understand controls to be some form of restraint, but how could such restraints be placed in the systems development process. Joe really had to talk to Randy.

That evening Joe called up Randy. It was always such a pleasure talking to him. He was indeed a walking bibliography. Joe was obviously very impressed with the depth and breadth of Randy’s knowledge. Randy pointed Joe to some of the earlier research undertaken by Richard Baskerville, a professor at Georgia State University. Randy also summarized the principle argument propounded in Baskerville’s book, *Designing Information System Security* [1]. Baskerville’s book was based on his doctoral research at the London School of Economics. In his book, Baskerville had argued that the only way security could be designed was by integrating it in software development process. Baskerville had argued that by introducing control transforms into conventional data flow diagrams ensured that security was being taken care of at the conceptualization stage.

Joe liked what he understood from Randy’s description. He had to buy the book.

About the same time Joe was introduced to CRAMM. The IT manager at SureSteel had mentioned that CRAMM was the way to go if they had to manage their risks. Apart from this introduction Joe really did not have any idea as to what CRAMM was and what it stood for. Even if he did, he didn’t remember anything. In attempts to finding out more about CRAMM, he reverted to the Internet and wrote CRAMM in the google.com search engine. His first hit was the cramm.com site. It was some consultant, Insight Consulting that showcased the product. The name of the firm sounded very familiar to Joe. He tried to remember where he had heard of Insight. Then he reached for
his visiting card box to dig into the number of visiting cards that numerous people had shared with him. There it was…Ian Glover, Insight Consulting. On the back Joe had scribbled “93/94 session”. Aah!... exclaimed Joe. He had met Ian when he had presented a seminar on risk management in London sometime in 1993/94. They had exchanged cards then. CRAMM was in its making then through the British technology office – CCTA. And CRAMM was really the CCTA Risk Assessment and Management Methodology. This was interesting.

As Joe read Baskerville and CRAMM, it occurred to him that indeed some of Baskerville’s thoughts were echoed in CRAMM usage, especially the relationship between CRAMM and systems development. CRAMM was being positioned as the risk management methodology alongside structured systems development methodologies. The more Joe dwelled into it the more he realized how much more he wanted to know…or rather how little he knew.

________________________________

In the ever increasing organizational complexity and the growing power of information technology systems, proper security controls have never been more important. As the modern organization relies on information technology based solutions to support its value chain and to provide strategic advantages over its competitors, the systems that manage those data must be regarded as one of its most valuable assets. Naturally, as an asset of an organization, information technology solutions must be safeguarded against unauthorized use and/or theft, and therefore, proper security measures must be taken.

This chapter provides a brief background of the causes of security breaches as they relate to system development, provide an explanation of common control methods and tools available for systems development, and provide a structured approach to achieve steady improvements within the systems development cycle of an organization. This chapter will not concentrate on specific “how-to” security procedures and technical
constructs, but will instead concentrate on conceptual frameworks that can be generalized to an arbitrary system architecture.

Security Breaches in Systems Development

Information systems textbooks regularly refer to the four types of access that may be granted to a database: create, read, update, and delete (CRUD). Unauthorized or accidental CRUD access also represents the four types of security breaches that may occur in systems that deal with data. The same type of unauthorized access may also be extended to applications, with the addition that applications may be executed.

Conceptually, no system should allow unauthorized access to data and/or applications. Therefore, all security breaches, without exception, are the result of a system failure. System failures fall into one of three categories [2]:

- Failure to perform a function that should have been executed
- Performance of a function that should not have been executed
- Performance of a function that produced an incorrect result

It should be noted that most failures do not occur as the result of a function being performed contrary to its design. Most processes occur exactly as they were designed to do. Therefore, most failures are the result of at least one design flaw. As long as a system performs any function, it is not possible to completely eliminate the potential for a system failure; therefore it is only possible to reduce the potential of an event. Control structures are the main tools with which to minimize the three types of system failures. As most system failures carry a cost (i.e. direct costs to correct the failure, lost productivity, lost good will, or financial penalties), an organization should implement control structures to prevent the occurrence of a system failure. However, it is also essential to implement control structures that will detect a failure: it is not possible for an organization to correct a failure if the organization does not know that it has occurred (except by accident).
Control Structures

There are various kinds of control structures that are essential to take note of in any systems development process. In this section these have been classified these into four categories: Auditing, Application controls, Modelling controls and Documentation controls.

Auditing

The process of auditing a system is one of the most fundamental control structures that may be used to examine, verify, and correct the overall functionality of that system. In order to audit an event, that event must necessarily have occurred. Fundamentally, the audit process verifies that a system is performing the functions that it should. This verification is of critical importance to security processes. It is not sufficient merely to implement security procedures: the organization must have a way to verify that the security procedures work. No manager or IT professional should be satisfied to say “we have not had a security breach; therefore our procedures are adequate.”

The audit process can be accomplished in two primary forms. Although both forms are actually variations of the same concept, most organizations distinguish the two for implementation purposes. The first form of the audit function is the record of changes of state in a system: i.e. events and/or changes in the system are recorded. The second form of the audit function is a systematic process that examines and verifies a system: i.e. the system is evaluated to determine if it is functioning correctly. In order for the audit control structure to be successful, both forms of the audit control structure need to be in place. It is difficult to perform a systematic evaluation of a system or event if they are not recorded. Furthermore, if changes of state are not recorded, there is no reason to record the changes. Therefore, successful auditing control procedures should:

- Record the state of a system
- Examine, verify, and correct the recorded states

All too often, the audit function within an organization is invoked only during the production stage of a process. In reality, the audit process should be invoked at all
stages of the system life-cycle. Generally speaking, the costs associated with the correction of an error in a system increase as the system life-cycle progresses. While the word “audit” conjures images of a Certified Public Accountant combing through files looking for wrong-doing within the organization, the scope of the audit process encompasses every aspect of a system. Clearly, this could become prohibitively expensive if independent auditors were to verify every aspect of and every event within a system. Therefore, who should be performing the audit function? Everyone should be performing the audit function. It is the responsibility of everyone working on a system, whether during design or production, to think critically and ask “is the system doing what it is supposed to be doing?”

**Application Controls**

In most general terms, application controls are the set of functions within a system that attempt to prevent a system failure from occurring. Application controls address three general system requirements:

- Accuracy
- Completeness
- Security

Accuracy and completeness controls both address the concept of correctness: i.e. that the functions performed within a system return the correct result. Specifically, accuracy controls address the need for a function to perform the correct process logic, while completeness controls address the need for a function to perform the process logic on all of the necessary data. Finally, security controls attempt to prevent the types of security breaches previously discussed.

Application controls are categorized based on their location within a system, and can be classified as either:

- Input controls
- Processing controls
• Output controls

Although many system failures are the result of input error, processing logic can be incorrect at any point within the system. Therefore, in order to meet the general system requirements stated above, all three classes of system controls are necessary in order to minimize the occurrence of a system failure.

While security controls often concentrate on the prevention of intentional security breaches, most breaches are accidental. As authorized users interact with a system on a regular basis, the likelihood of accidental breaches is much higher than deliberate breaches; therefore the potential cost of accidental breaches are also much higher.

Application controls are typically considered to be the passwords and data integrity checks imbedded within a production system. However, application controls should be incorporated at every stage of the system life cycle. In the highly automated development environments of today, application controls are just as necessary to protect the integrity of system development and integration: for example, the introduction of malicious code at the implementation stage could quite easily create costly system failures once a system were put into production. In order to best minimize the occurrence of a system failure, application controls and audit controls should be coordinated as part of a comprehensive strategy covering all stages of the system life cycle. Application controls address the prevention of a failure, and audit controls address the detection of a failure, i.e. audit controls attempt to determine if the application controls are adequate.

**Modelling Controls**

Modelling controls are used at the analysis and design stages of the systems life cycle as a tool to understand and document other control points within a system. Modelling controls allow for the incorporation of audit and application controls as an integral part of the systems process, rather than relying on the incorporation of controls as add-on functionality. Just as with other modelling processes within a system, modelling controls will take the form of both logical and physical models: logical
controls illustrate controls required as a result of business rules, and physical controls illustrate controls required as a result of the implementation strategy.

As an example, prudence dictates that an on-line banking system would necessarily require security. However, it is not sufficient to simply “understand” during the initial system development that security would be required. Modelling controls show how the control would interact with the overall functionality of the system, and locates the control points within the system. In the on-line banking system, the logical control model might include a control point that “authenticates users.” The implementation control model might include a control point “verify user id and password.”

In addition to the inclusion of the control point, the model should demonstrate system functionality should the tests of the control point fail. In the on-line banking example above, if the authentication test fails, will the user simply be allowed to try an indefinite number of times? Will the user account be locked after a certain number of failed attempts? Will failed attempts be logged? Will the logs be audited to search for potential security threats? The answers to all of these questions should be included and modelled from the first stages of the system development process.

Consider the example of a user accessing the account online. Figure 5.1 illustrates the situation where there are no controls. Any user authentication system however is an example of a control point (Figure 5.2). The system with proper controls however would include aspects of account getting locked if the user is not authenticated, presenting instructions once the account is locked out and granting access if the user is authenticated. The full set of controls is shown in Figure 5.3.
Figure 5.1, Model without controls

Figure 5.2, Model with control point
Figure 5.3. Model with full controls included
Documentation Controls

Documentation is one of the most critical controls that can be used to maintain integrity within a system; ironically, it is also one of the most neglected. Documentation should exist for all stages of the system life cycle, from initial analysis through maintenance. In theory, a system should be able to be understood from the documentation alone, without requiring study of the system itself. Unfortunately, many IT professionals consider documentation to be secondary to the actual building of the system itself – to be done after the particular development activity has been completed. After all, the most important result is the construction of the system!

In reality, documentation should be created in conjunction with the system itself. Although the documentation of results after a particular phase is important, document controls should be in place before, during, and after that phase. Furthermore, while it is true that documentation is a commitment of resources that could be used toward other development activities, proper and complete documentation can ultimately save resources by making a system easier to understand. This savings is particularly true during the production and maintenance stages. How many programmers have had to reverse engineer a section of program code in order to learn what the code is supposed to be doing? Good documentation dramatically increases the accuracy and reliability of other controls, such as auditing. In the previous example, by already knowing the purpose of a section of a code, the programmer could spend his time verifying that the code was performing its intended purpose.

Good documentation controls not only answer what are the functions of a system and how those functions are being accomplished, the controls address the question of why the system is performing those particular functions. Specifically, documentation should show the correlation between system functionality and business/implementation needs. For example, an application control may be placed within a system in order to meet a specific requirement. If that requirement were to change, the control may no longer be needed and would become a waste of resources. Worse yet, the control may actually conflict with new requirements.
Process Improvement Software

As proper controls are a requirement in order to foster improvement in organizational systems, there are tools available to assist with the successful implementation of controls within the development strategy. In addition, these automated solutions can assist with a general environment of improvement within the software development life cycle. Software tools would include classes such as automated learning and discover tools, which assist with the analysis of data and program structures to help determine exactly what is happening within a system. A second class of automated improvement tools is the Program Enhancement Environment. These software environments assist with the management of the software life cycle from beginning to end; rather than using different tools at each stage, a single development suite can integrate multiple stages into a comprehensive package.

As was discussed, proper documentation is one of the most critical controls that can be integrated into the system development cycle. Managing and tracking documentation within large systems can be a daunting task. Two classes of tracking software can provide invaluable assistance in large systems development processes: change tracking software and requirements tracking software. As its name implies, change tracking software tracks changes through the development process. Although its purpose is relatively straight-forward, the benefits are gained by the sheer volume of information that can be managed.

All systems development projects are designed to meet certain business requirements. Through the analysis and design phases, these business processes are translated and modelled into logical and physical constructs that function to satisfy the requirement. During implementation, these models are transformed into data stores, classes, and procedures. Requirements tracking is the process of connecting business requirements, through analysis and design, to the program artifacts that support the requirement. Essentially, requirements tracking is a process that proves that a system meets the business requirements for which it was designed. Although not all system development projects require the rigors of requirements tracking, the process can prove valuable. Furthermore, requirements tracking can be a condition placed upon contractors
performing development services: the contractor may not receive the bid if the tracking is not performed, or the contractor may not be compensated if satisfactory tracking is not demonstrated. As systems grow in size and complexity, it can become impossible to maintain these connections manually, and requirements tracking software may become the only feasible solution.

The SSE-CMM

Along with our discussion about controls, we would like to introduce the Systems System Security Engineering Capability Maturity Model (SSE-CMM). This methodology follows the theme of our topic. The System Security Engineering Capability Maturity Model (SSE-CMM) is a model used for assessing the security engineering aspects of the target organization. A crucial reason that organizations need it is for confidence. It provides confidence that an organization can do what it claims to do. Also, developmental assurance is also achieved. Essentially this follows the concept of something being on time and within budget. The SSE-CMM is a metric for determining the best candidate for a specified security activity. The first public workshop on SSE-CMM was conducted during the first quarter of 1995. At that time the definition of the concept was introduced. Version 1.0 of Model and System Security Appraisal Methodology (SSAM) was released in October 1996. Initially five pilots were carried out with great success. The second public workshop was carried out during the summer of 1997 where version 1.1 of the model and SSAM was also introduced. The results of the pilots validated the use of the model for internal assessment.

What is the SSE-CMM?

The SSE-CMM is a model, which is based upon the requirements for implementing security in systems or a series of such related systems, which lie in the domain of Information Technology Security (ITS). The SSE-CMM model could be defined in various ways. Rather than invent an additional definition, the SSE-CMM Project chose to adapt the definition of systems engineering from the Software Engineering Capability Maturity Model as follows:
“Systems security engineering is the selective application of scientific and engineering efforts to:
transform a security policy statement into a description of a system that best satisfies the security
policy according to accepted measures of effectiveness (e.g., functional capabilities) and need for
assurance; integrate related security parameters and ensure compatibility of all environmental,
administrative, and technical security disciplines in a manner which optimizes the total system
security design; integrate the system security engineering efforts into the total system engineering
effort” (System Security Engineering Capability Model Description Document, Version 3.0,

It addresses a special area called system security and SSE-CMM is designed
using the generalized framework provided by the Systems Engineering CMM as a
foundation (See Figure 5.4 for CMM levels). The model architecture separates the
specialty domain from process capability. In the case of SSE-CMM, it is a specialty
domain with system security engineering process areas separated from the generic
characteristics of the capability side. Here the generic characteristics relate to increasing
process capability.

Figure 5.4. CMM Levels

A question that is often asked is that why is security engineering important?
Clearly information plays an important role in shaping up the way business is being
conducted in this era of the Internet. Information is an asset that has to be properly
deployed to get the maximum benefits out of it. From mundane day-to-day operational
decisions, information can be used to provide strategic directions to the corporation. Thus not only acquiring the relevant data is important but also the security of the vital data acquired is also an issue of paramount concern. Many systems, products and services are needed to maintain and protect information. The focus of security engineering has expanded the horizons of the need for data protection, hence security, from classified government data to broader applications including financial transactions, contractual agreements, personal information and the Internet. These trends have increased the need for security engineering and by all probabilities these trends seem to be there to stay.

Within the ITS domain the SSE-CMM Model is focused on processes which can be used in achieving security and the maturity of these processes. It does not show any specific process or way of doing particular things, rather it expects organizations to base its processes in compliance with any ITS guidance document. The scope of these processes should incorporate the following:

1. System security engineering activities used for a secure product or a trusted system. It should address the complete lifecycle of the product, which includes:
   a. conception of idea
   b. requirement analysis for the project
   c. designing of the phases
   d. development, integration of the parts
   e. proper installation
   f. operation and maintenance.

2. Requirements for the developers (product and secure system) and integrators, the organizations that provide computer security services and computer security engineering.
3. It should be applicable to various companies that deal with security engineering, academia and government.

SSE-CMM promotes the integration of various disciplines of engineering, as security is an issue, which is important and useful to all the engineering disciplines.

**Need for the Model**

Why was SSE-CMM developed in the first place? Why was the need to have a reference model like SSE-CMM felt? When we venture into the context of the development of this type, we realize that there could be various reasons that called for this kind of an effort. Every business is interested in increasing efficiency, a practical way for which could be to have a process that provides a high quality product with minimal cost. Most statistical process controls suggest that higher quality products can be produced most cost-effectively by emphasizing on the quality of processes that produce them, and the maturity of the organizational practices inherent in these processes. More efficient processes are warranted, given the increasing cost and time required for the development of secure systems and reliable products. These factors again can be linked to people who manage the technologies.

As a response to the problems identified above, the SEI (Software Engineering Institute) began developing a process maturity framework. This framework would help organizations improve their software processes and guide them in becoming mature organizations. A mature software organization possesses an organization-wide ability for managing software development process. The software process is accurately communicated to the staff and work activities are carried out according to the planned process. A disciplined process is consistently followed and always ends up giving better quality controls as all of the participants understand the value of doing so, and the necessary infrastructure exists to support the process.

Initially, the SEI released a description of the framework along with a maturity questionnaire. The questionnaire provided the tool for identifying areas where an organization's software process needed improvement. The initial framework has evolved over a period of time because of ongoing feedback from the software community, into
the current version of the SEI CMM for software, which is now being implemented. The SEI CMM describes a model of incremental process improvement. It provides organizations with a sequence of process improvement levels called Maturity Levels. Each Maturity Level is characterized by a set of software management practices. Each level provides a foundation to which the practices of the next level are added. Hence the sequence of levels defines a process of incremental maturity. The primary focus of the SEI CMM is the management and organizational aspects of software engineering. The idea is to develop an organizational culture of continuous process improvement. After years of assessment and capability evaluations using SEI CMM it benefits are being realized today.

Results from implementation of the SEI CMM concepts indicate that improved product quality and predictable performance can be achieved by focusing on process improvement. Long-term software industry benefits have been as good as ten-fold improvement in productivity and one-hundred-fold improvement in quality. The return on investment (ROI) of process improvement efforts is also high. The architecture of SSE-CMM was adopted from CMM since it supports the use of process capability criteria for specialty domain areas such as system security engineering.

The objective of the SSE-CMM Project has been to advance the security-engineering field. It helps the discipline to be viewed as mature, measurable and defined. The SSE-CMM model and appraisal methods are being developed to:

1. Help in making investments in security engineering tools, training, process definition, and management practice worthwhile. It helps in improvements by engineering groups.

2. Help in providing capability-based assurance. Trustworthiness is increased based on confidence in the maturity of an engineering group’s security and practices.

3. Help in selecting appropriately qualified providers of security engineering through differentiating bidders by capability levels and associated programmatic risks.
**History of SSE-CMM Project**

The SSE-CMM initiative began as a National Security Agency (NSA) sponsored effort in April 1993 with research into existing work on Capability Maturity Models (CMMs) and investigation of the need for a specialized CMM to address security engineering. During this early Phase, a ‘strawman’ Security Engineering CMM was developed to match the requirement. The information security community was invited to participate in the effort at the First Public Security Engineering CMM Workshop in January 1995. Representatives from over 60 organizations reaffirmed the need for such a model. As a result of the community’s interest, Project Working Groups were formed at the workshop, initiating the Develop Phase of the effort. The first meeting of the working groups was held in March 1995. Development of the model and appraisal method was accomplished through the work of the SSE-CMM Steering, Author, and Application Working Groups with the first version of the model published in October 1996 and of the appraisal method in April 1997.

To validate the model and appraisal method, pilots occurred from June 1996 through June 1997. These pilots provided valuable input to Version 1.1 of the model and appraisal method. The first version of the model was used in pilots that appraised two large system integration efforts, two service providers, and a product developer. The pilots addressed various organizational aspects that contributed to the validation of the model, including:

- Organizations of various sizes
- Contract-driven system development and market-driven product development
- High and low assurance developments
- Development
- Operational, and
- Service provider organizations
In July 1997, the Second Public Systems Security Engineering CMM Workshop was conducted to address issues relating to the application of the model, particularly in the areas of acquisition, process improvement, and product and system assurance. As a result of issues identified at the workshop, new Project Working Groups were formed to directly address the issues. Subsequent to the completion of the project and the publication of version 2 of the model, the International Systems Security Engineering Association (ISSEA) was formed to continue the development and promotion of the SSE-CMM. In addition, ISSEA took on the development of additional supporting materials for the SSE-CMM and other related projects. ISSEA continues to maintain the model and its associated materials as well as other activities related to systems security engineering and security in general. ISSEA has become active in the International Organization for Standardization and sponsored the SSE-CMM as an international standard ISO/IEC 21827.

**Description of SSE-CMM Project Composition**

As is identified in figure 5.5, the project consists of the following groups:

**The Steering Group**: This is the top most group acting as the project leaders. It provides oversight and guidance for the SSE-CMM work processes, products, and progress. There work is also to encourage the acceptance and adoption of the SSE-CMM.

**The Appraisal Method Group**: This group is primarily responsible for maintaining the SSE-CMM Appraisal Method (SSAM), including the development of a third-party appraisal method. It not only plans and supports a pilot trial program but also analyzes a pilot trial program for testing the third party appraisal method, if necessary.

**The Model Maintenance Group**: The responsibility of this group lies in maintenance of the model. This includes ensuring that the process areas cover all security activities in all communities, minimizing conflicts of SSE-CMM with other models. It also accurately articulates the SSE-CMM's relationship to other efforts in the model document.
The Life Cycle Support Group: This group being at the same level in the structure, is responsible for developing and establishing a mechanism for appraiser qualification and appraisal team comparability, designing and implementing a repository for the maintenance of assessment data, and preparing and issuing guidance on rating interpretation and rating maintenance.

The Profiles, Assurance, and Metrics Group: This group works a level below the above groups and is responsible to investigate and validate the concept of profiles, determine and document the contribution of the SSE-CMM to assurance. It also attempts to find out and validate security and process metrics relating to the use of the SSE-CMM.

The Sponsorship, Planning and Adoption Group: This group works at the same level as the above group and is responsible for pursuing sponsorship options. It defines and plans for the organization to maintain the SSE-CMM. Other responsibilities include developing and maintaining the integrated Project Schedule, and promoting and pursuing use/adoption of the SSE-CMM in various communities of interest.
**Key Reviewers:** These are the people who make a formal commitment to review and provide timely comments on SSE-CMM Project work products.

**Community Reviewers:** These people may also review work products but without formal commitment.

The SSE-CMM was developed by the collaboration of a group of companies with long and successful histories in building secure products and systems, and/or in the provision of secure services. It has a tradition of incorporating the community feedbacks from these people. Key Reviewers, selected from various backgrounds for their security engineering expertise, supplement the principal authors. The authors also incorporated feedback from the 1st public workshop where an early version of the model was critiqued. Ongoing development and maintenance efforts continue to solicit and incorporate community feedback in subsequent version of the model.

A consensus process developed the SSE-CMM model. All member organizations could send representatives to the working group meetings, and the majority did. Contributions were sent electronically to all members of the working group in the intervening period between meetings. Meetings were held on a monthly basis where input suggestions were discussed, revised and agreed. The results of any votes that were necessary were recorded in the working group meeting minutes issued for each meeting. These records have been maintained.

The working group tasked with development, first approved each version of the SSE-CMM Model. It was then reviewed and approved by the Steering Group. After the Steering Group had approved the version it was then sent to a group of “Key reviewers” drawn from the ITS community at large for their review and comment. Each version was then released for public review and feedback. These feedbacks from the Key reviewers and the community at large helped the Steering Group to make a final decision regarding the release of that version of the SSE-CMM Model. The SSE-CMM Model has been approved first at the working group level; second at the Steering Group level, third at the Key Reviewer level and finally at the community level. Thus four levels of approval have been obtained before actual release of the version.
Additional approval and consensus has been achieved during the Pilot Appraisals through the impact of application of the Model to different application domains. The Alternative Assurance Working Group (AAWG) of the Common Criteria Project has reviewed the SSE-CMM Model for applicability as an alternative to the generation of assurance by evaluation and provided IT systems security community consensus feedback to the project.

CMM: Constructs and Concepts

Process is one of the major determinants of cost and quality. Thus, ways to improve processes is a major concern for companies. Organizational potential decides the process capability. Similarly another concept, process maturity, defines the extent to which a particular process is explicitly defined, managed, controlled and effective. Using CMM framework, an engineering organization, can turn from less organized into a highly structured and effective organization. The SSE-CMM model was developed with the anticipation that applying the concepts of statistical process control to security engineering is going to promote secure system development within limits of cost, schedule and quality.

In this section, concepts are presented and explained in the context of SSE-CMM model, which allow a better understanding of the model. Some of the key concepts involved with SSE-CMM model are presented below:

Organizations and Projects

To differentiate between two aspects of organizational structure, both organization and project have been separately defined. Other constructs such as teams, exist within business entities, but there is no commonly accepted terminology that spans all business contexts. These two terms were chosen because they are usually interpreted in ambiguous ways by most of the targeted audience of the SSE-CMM.

Organization. To implement SSE-CMM at the organizational level, it becomes important to define the term “organization”. For this purposes, an organization may be defined as a unit or subunit within a company, the whole company or any other entity like a government institution or service utility, responsible for the oversight of multiple
projects. All projects within an organization typically share common policies at the top of the reporting structure. An organization may consist of geographically distributed projects and supporting infrastructures. The term “organization” is used to connote an infrastructure to support common strategic, business, and process-related functions. The infrastructure exists and must be utilized and improved for the business to be effective in producing, delivering, supporting, and marketing its products.

**Project.** The project in context of SSE-CMM may be defined as the aggregate of effort and other resources focused on developing and/or maintaining a specific product or providing a service. The product may include hardware, software, and other components. Typically a project has its own funding, cost accounting, and delivery schedule. A project may constitute an organizational entity completely on its own. It could also constitute a structured team, task force, or other group used by the organization to produce products or provide services. The categories of organization and project are distinguished based on typical ownership. In SSE-CMM definition, one could differentiate between project and organization categories by defining the project as focused on a specific product, whereas the organization encompasses one or more projects.

**System**

In SSE-CMM, system refers to an integrated composite of people, products, services, and processes that provide a capability to satisfy a need or objective. It can also be viewed as an assembly of things or parts forming a complex or unitary whole (i.e., a collection of components organized to accomplish a specific function or set of functions).

A system may be a product that is hardware only, hardware/software and software only, or a service. The term “system” is used throughout the model to indicate the sum of the products being delivered to the customer(s) or user(s). In SSE-CMM, a product is denoted a system to emphasize the fact that we need to treat all the elements of the product and their interfaces in a disciplined and systematic way, so as to achieve the overall cost, schedule, and performance (including security) objectives of the business entity developing the product.
Work Product

Anything generated in the course of performing a process of the organization could be termed as a “work product”. These could be the documents, the reports generated during a process, the files created, the data gathered or used, etc. Here rather than listing the individual work products for each process area, SSE-CMM lists “Example Work Products” of a particular base practice, as it can elaborate further the intended scope of a base practice. These lists are illustrative only and reflect a range of organizational and product contexts.

Customer

A customer, as defined in context of the model, is the entity (individual, group of individuals, organization) for whom a product is developed or service is made, or the entity (individual, group of individuals, organizations) that uses the product or service. The usage of customer in SSE-CMM context has an implication of understanding the importance of the users of the product, to target the right segment of consumers of the product.

In the context of the SSE-CMM, a customer may be either negotiated or non-negotiated. A negotiated customer is entity who contracts with another entity to produce a specific product or set of products according to a set of specifications provided by the customer. A non-negotiated, or market-driven, customer is one of many individuals or business entities who have a real or perceived need for a product.

In the SSE-CMM model, the individual or entity using the product or service is also included in the notion of customer. This is relevant in the case of negotiated customers, since the entity to which the product is delivered is not always the entity or individual that will actually use the product or service. It is the responsibility of the developers (at supply side) to attend to the entire concept of customer, including the users.

Process

Several types of processes are mentioned in the SSE-CMM, some of which could be “defined” or “performed” processes. A defined process is formally described for or by
an organization for use by its security engineers. The defined process is what is expected of the organization's security engineers to do. The performed process is what these people (security engineers) actually end up doing.

If a set of activities is performed to arrive at an expected set of results, then it can be defined as a “process”. Activities may be performed iteratively, recursively, and/or concurrently. Some activities can transform input work products into output work products needed. The allowable sequence for performing activities is constrained by the availability of input work products and resources, and by management control. A well-defined process includes activities, input and output artifacts of each activity, and mechanisms to control performance of the activities.

**Process Area**

A process area (PA) can be defined as a group of related security engineering process characteristics, which when performed in a collective manner, can achieve a defined purpose. It is composed of base practices, which are mandatory characteristics that must exist within an implemented security engineering process before an organization can claim satisfaction in a given process area. SSE-CMM identifies 10 process areas. These are: Administer Security Controls, Assess Operational Security Risk, Attack Security, Build Assurance Argument, Coordinate Security, Determine Security Vulnerabilities, Monitor System Security Posture, Provide Security Input, Specify Security Needs, Verify and Validate Security. Each process area has predefined goals. SSE-CMM process areas and goals appear in table 5.1.
Table 5.1. SSE-CMM Security Engineering Process Areas (from [3]).

<table>
<thead>
<tr>
<th>Process Area</th>
<th>Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administer Security Controls</td>
<td>- Security controls are properly configured and used.</td>
</tr>
<tr>
<td>Assess Operational Security</td>
<td>- An understanding of the security risk associated with operating the</td>
</tr>
<tr>
<td>Risk</td>
<td>- system within a defined environment is reached.</td>
</tr>
<tr>
<td>Attack Security</td>
<td>- System vulnerabilities are identified and their potential for</td>
</tr>
<tr>
<td></td>
<td>- exploitation is determined.</td>
</tr>
<tr>
<td>Build Assurance Argument</td>
<td>- The work products and processes clearly provide the evidence that</td>
</tr>
<tr>
<td></td>
<td>- the customer’s security needs have been met.</td>
</tr>
<tr>
<td>Coordinate Security</td>
<td>- All members of the project team are aware of and involved with</td>
</tr>
<tr>
<td></td>
<td>- security engineering activities to the extent necessary to perform</td>
</tr>
<tr>
<td></td>
<td>- their functions.</td>
</tr>
<tr>
<td></td>
<td>- Decisions and recommendations related to security are communicated</td>
</tr>
<tr>
<td></td>
<td>- and coordinated.</td>
</tr>
<tr>
<td>Determine Security</td>
<td>- An understanding of system security vulnerabilities is reached.</td>
</tr>
<tr>
<td>Vulnerabilities</td>
<td></td>
</tr>
<tr>
<td>Monitor System Security</td>
<td>- Both internal and external security related events are detected and</td>
</tr>
<tr>
<td>Posture</td>
<td>- tracked.</td>
</tr>
<tr>
<td></td>
<td>- Incidents are responded to in accordance with policy.</td>
</tr>
<tr>
<td></td>
<td>- Changes to the operational security posture are identified and</td>
</tr>
<tr>
<td></td>
<td>- handled in accordance with security objectives.</td>
</tr>
<tr>
<td>Provide Security Input</td>
<td>- All system issues are reviewed for security implications and are</td>
</tr>
<tr>
<td></td>
<td>- resolved in accordance with security goals.</td>
</tr>
<tr>
<td></td>
<td>- All members of the project team have an understanding of security so</td>
</tr>
<tr>
<td></td>
<td>- they can perform their functions.</td>
</tr>
<tr>
<td></td>
<td>- The solution reflects the security input provided.</td>
</tr>
<tr>
<td>Specify Security Needs</td>
<td>- A common understanding of security needs is reached between all</td>
</tr>
<tr>
<td></td>
<td>- applicable parties, including the customer.</td>
</tr>
<tr>
<td>Verify and Validate Security</td>
<td>- Solutions meet security requirements.</td>
</tr>
<tr>
<td></td>
<td>- Solutions meet the customer’s operational security needs.</td>
</tr>
</tbody>
</table>

**Role Independence**

When the process areas of the SSE-CMM are joined together as groups of practices and taken together, it achieves a common purpose. But, the groupings are not meant to imply that all base practices of a process are necessarily performed by a single individual or role. This is one way in which the syntax of the model supports the use of it across a wide spectrum of organizational contexts.

**Process Capability**

Process capability is defined as the range (which is quantifiable) of results that are expected or can be achieved by following a process. The SSE-CMM Appraisal Method (SSAM) is based on statistical process control concepts that define the use of process capability. The SSAM can be used to determine process capability levels for
each process area within a project or organization. The capability side of the SSE-CMM reflects these concepts and provides guidance in improving the process capability of the security engineering practices that are referenced on the domain side of the SSE-CMM.

The capability of an organization's process is instrumental in predicting the ability of a project to meet goals. Projects in low capability organizations experience wide variations in achieving cost, schedule, functionality, and quality targets.

Institutionalization

Institutionalization is the building of infrastructure and corporate culture that establishes methods, practices, and procedures. These established practices stay even after those who originally defined them are gone. The process capability side of the SSE-CMM supports institutionalization by providing a path and offering practices toward quantitative management and continuous improvement. In this way the SSE-CMM asserts that organizations need to explicitly support process definition, management, and improvement. Institutionalization provides a means to gain maximum benefit from a process that exhibits sound security engineering characteristics.

Process Management

Process management is the management of related set of activities and infrastructures, which are used to predict, then evaluate, and finally control the performance of a process. Process management implies that a process is defined (since one cannot predict or control something that is undefined). The focus on process management implies that a project or organization takes into account all possible factors regarding both product and process related problems in the planning phase, at performance level, in evaluating and monitoring, and also corrective action.

Capability Maturity Model

A capability maturity model (CMM) such as the SSE-CMM describes the stages through which processes show progress as they are defined initially, implemented practically, and improved gradually. The model provides a way to select process improvement strategies by firstly determining the current capabilities of specific processes and then subsequently identifying the issues most critical to quality and
process improvement within a particular domain. A CMM may take the form of a reference model to be used as a guide for developing and improving a mature and defined process.

A CMM may also be used for appraisal of the existence of a process and institutionalization of a defined process that implements referenced practices. A capability maturity model covers all the processes that are used to perform the tasks of the specified domain, (e.g., security engineering). A CMM can also cover processes used to ensure effective development and use of human resources, as well as the insertion of appropriate technology into products and tools used to produce them.

**SSE-CMM Architecture Description**

The SSE-CMM architecture is designed to enable a determination of a security engineering organization’s process maturity across the breadth of security engineering. The model evaluates each process area against common feature. The goal of the architecture is to separate basic characteristics of the security engineering process from its management characteristics. In order to ensure this separation, the model has two dimensions, called “domain” and “capability” (described below). Importantly, the SSE-CMM does not imply that any particular group or role within an organization must undertake any of the processes described in the model. Nor does it require that the latest and greatest security engineering technique or methodology be used. The model does require, however, that an organization have a process in place that includes the basic security practices described in the model. The organization is free to create their own process and organizational structure in any way that meets its business objectives. The generic levels of SSE-CMM appear in Figure 5.6.
The Basic Model

The SSE-CMM has two dimensions.

1. **Domain**: This consists of all the practices, that together in, a collective manner define security engineering in an organization. These practices could be called the “base practices”.

2. **Capability**: This represents practices that indicate process management and institutionalization capability. These are also known as “generic practices” as they apply across a wide range of domains.

Base Practices

The SSE-CMM contains 129 base practices, organized into 22 process areas. Of these 61 base practices, organized in 11 process areas, cover all major areas of security engineering. The remaining 68 base practices, organized in 11 process areas, address the project and organization domains. They have been drawn from the Systems Engineering and Software CMM. They are required to provide a context and support for the Systems Security Engineering process areas.
The base practices for security were gathered from a wide range of existing materials, practice, and expertise. The practices selected represent the best existing practice of the security engineering community, but not untested practices.

To identify security engineering base practices is a complicated task as there are several names for the same activities. These activities could occur later in the life cycle, at a different level of abstraction, or individuals in different roles could perform them. However, an organization cannot be considered to have achieved a base practice if it is only performed during the design phase or at a single level of abstraction. Therefore, the SSE-CMM has ignored these distinctions and tries to identify the basic set of practices that are essential to the practice of good security engineering.

Thus a base practice can have the following characteristics:

- Should be applied across the life cycle of the enterprise
- Should not overlap with other Base Practices
- Should represent a “best practice” of the security community
- Should be applicable using multiple methods in multiple business contexts
- Should not specify a particular method or tool

The base practices have been organized into process areas such that it meets a broad spectrum of security engineering organizations. There are many ways to divide the security-engineering domain into process areas. One might try to model the real world or creating process areas that match security-engineering services. Other strategies attempt to identify conceptual areas that form fundamental security engineering building blocks.

**Generic Practices**

Generic practices are activities by definition that should be applicable to all processes. They address all the aspects of the process: management, measurement and institutionalization. They are used for an initial appraisal, which helps in determining the capability of an organization to perform a particular process. Generic practices are
grouped into logical areas called “Common Features” which are organized into five “Capability Levels”, which represent increasing organizational capability. Unlike the base practices of the domain dimension, the generic practices of the capability dimension are ordered according to maturity. Therefore, generic practices that indicate higher levels of process capability are located at top of the capability dimension.

The common features here are designed in a way such that it helps in describing major shifts in an organization's manner of performing work processes (in this case, the security engineering domain). Each common feature has to have one or more generic practices. Subsequent common features have generic practices, which helps in determining or assessing how well a project manages and improves each process area as a whole.

**The Capability Levels**

The way in which the common features are ordered can be derived from the observation that implementation and institutionalization of some practices benefit from the presence of other practices. This is especially more applicable if practices are well established. Before an organization can define, tailor, and use a process effectively, individual projects should have some experience managing the performance of that process. Before institutionalizing a specific estimation process across the entire organization, for example, an organization should at least, first attempt to use the estimation process on a project. However, some aspects of process implementation and institutionalization should be considered together (not one ordered before the other) since they work together towards enhancing capability.

Common features and capability levels are important both in performing an assessment of the current processes and improving an organization's process capability. In the case of an assessment where an organization has some, but not all common features implemented at a particular capability level for a particular process, it usually operates at the lowest completed capability level for that process. An organization may not reap the full benefit of having implemented a common feature if it is in place, but not all common features at lower capability levels. An assessment team should take this into account in assessing an organization's individual processes. In the case of improvement,
organizing the practices into capability levels provides an organization with an “improvement road map,” should it desire to enhance its capability for a specific process. For these reasons, the practices in the SSE-CMM are grouped into common features, which are ordered by capability levels.

An assessment should be performed to determine the capability levels for each of the process areas. This indicates that different process areas can and probably will exist at different levels of capability. The organization will then be able to use this process-specific information as a means to focus improvements to its processes. The priority and sequence of the organization's activities to improve its processes should take into account its business goals.
In brief

- Security related with flawed systems development typically occurs because of:
  - Failure to perform a function that should have been executed,
  - Performance of a function that should not have been executed, or
  - Performance of a function that produced an incorrect result.
- There are four categories of control structures: Auditing, Application controls, Modelling controls and Documentation controls.
- Auditing controls record the state of a system, and examine, verify, and correct the recorded states.
- Application controls look for accuracy, completeness, and general security.
- Modelling controls look for correctness in system specification.
- Documentation controls stress the importance of documentation alongside systems development rather than as an afterthought.
- SSE-CMM focuses on processes which can be used in achieving security and the maturity of these processes.
- The scope of the processes incorporate:
  - System security engineering activities used for a secure product or a trusted system. It should address the complete lifecycle of the product, which includes: conception of idea; requirement analysis for the project; designing of the phases; development, integration of the parts; proper installation; operation and maintenance.
  - Requirements for the developers (product and secure system) and integrators, the organizations that provide computer security services and computer security engineering.
  - It should be applicable to various companies that deal with security engineering, academia and government.
- SSE-CMM Process Areas include: Administer Security Controls; Assess Operational Security Risk; Attack Security; Build Assurance Argument; Coordinate Security; Determine Security Vulnerabilities; Monitor System Security Posture; Provide Security Input; Specify Security Needs; Verify and Validate Security
- SSE-CMM has two basic dimensions: base practices and generic practices

Questions and Exercises

Discussion questions. These are based on a few topics from the chapter and are intentionally designed for a difference of opinion. These questions can best be used in a classroom or a seminar setting.
1. “SSE-CMM is the panacea of secure systems development.” Discuss

2. How does SSE-CMM ensure correctness of system specification leading to good system design? Is there a connection between good system design and security? If so, what is it? If not, give reasons for lack of such a relationship.

3. Establishing control structures in systems can best be achieved by focusing on requirement definitions and ensuring that controls get represented in basic data flows. Although such an assertion seems logical and commonsensical, identify and examine hurdles that usually prevent us from instituting such controls.

**Exercise.** Think of a fictitious software house developing software for mission critical applications. Develop measures to assess the level of maturity for each of the SSE-CMM levels. Suggest reasons as to why your measures should be adopted.

**References**

