International Journal of Economics and Finance
www.ccsenet.org/journal.html
Vol. 1, No. 1
February 2009

Estimating Software Cost with Security Risk Potential

Nur Atiqah Sia Abdullah
Faculty of Information Technology and Quantitative Sciences
University of Technology MARA
40450 Shah Alam, Selangor Darul Ehsan, Malaysia
Tel: 60-3-5521-1175 E-mail: szeyieng@yahoo.com; atiqah@tmsk.uitm.edu.my

Rusli Abdullah, Hasan Selamat, Azmi Jaafar
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
43300 Serdang, Selangor Darul Ehsan, Malaysia
Tel: 60-3-8946-6575 E-mail: {rusli, hasan, azmi}@fsktm.upm.edu.my

Kamaruzaman Jusoff (Corresponding author)
Yale University
Yale's Centre for Earth Observation
21 Sachem St, New Haven, CT 06511, USA
Tel: 203-432-1384 E-mail: jusoff.kamaruzaman@yale.edu

Abstract

Software houses are now keen to provide secure software as requested by customers' desire with respect to security and quality of their products especially related to the software costing estimation in the software development and implementation environment. Therefore, there is a need to identify the potential security risks while estimating the application cost. In this paper, we provide a list of potential security risks throughout the system development life cycle (SDLC). This list provides useful insights for software developers and practitioners in identifying security risks so that it can be encountered as security cost in the application.

Keywords: Security factor, Software cost estimation, Software metrics

1. Introduction

Software cost estimation (SCE) is very important to estimate the cost that involved in developing software. SCE can be vary depends on different models. But most of the cost estimation models do not consider the costs that invoke security while developing software. It is because security is often an afterthought when developing software and is often bolted on late in development or even during deployment or maintenance (Shanai *et al*, 2006). Currently, secure software development has gained momentum during the past couple of years and improvements have been made. Software houses are now keen to provide secure software as requested by customers' desire with respect to security and quality of their products (Jari, 2006).

Besides, a risk discovered late in the project lifecycle becomes a fire to fight. In examining some real-world experiences, project managers faced project "fires" when risks were not identified during the planning phase. These fires often resulted in added costs or negative consequences (Frederick, 2006). But, engineering security will substantially raise software – project cost and there has been wide variation in the amount of added cost estimated by different models (Colbert et. al., 2004). Therefore, there is a need to include secure software costing while estimating the application cost. Extended the existing software cost estimation technique is essential in order to calculate the cost that includes security in application.

2. Overview of Related Works

2.1. Parametric Software Cost Estimation Models

From two decades, many studies that comparing parametric models (Kemerer, 1987; Srinivasan *et. al*., 1995; Chulani *et.*

al., 1998; Briand *et. al*., 1999) for software cost estimation have been published. There are numbers of widely used parametric models (Symons, 1991; Albrecht, 1979; Boehm, 1981; COSMIC, 2007; Putnam, 1978) are reviewed, compared and evaluated from many aspects or factors.

Kemerer (Kemerer, 1987) conducted a well-known study reporting on the relative accuracy of four software cost estimation models includes Function Points (Albrecht, 1979), SLIM (Putnam, 1978), COCOMO (Boehm, 1981), and ESTIMACS, in being able to predict the actual costs of 15 completed software development projects. The results indicated that the models require significant calibration, and the study provided insight to the factors affecting modern software development productivity. Briand et al. (1998) examined the accuracy of data-driven software cost modeling based on certain criteria and compared organization-specific with multi-organization models. The study assessed and compared the models based on criteria such as ordinary least square regression, stepwise ANOVA, Classification and Regression Trees (CART) and analogy. This study urged the need for risk analysis to be an integrated part of software cost estimation procedures (Briand *et al*., 1998).

Colbert et al (Colbert *et. al*., 2004; Colbert *et. al*., 2006) have developed a model for costing secure software-intensive systems (COSECMO), which extended from COCOMO II. COSECMO is based on behavior-analysis activities, such as analyzing industry practices with respect to security, 149 Security Targets that registered on the National Information Assurance Partnership (NIAP) website and conducting preliminary surveys of experts in software development and in security. COSECMO introduces a new cost driver (Wu *et. al*., 2006) and defines guides for setting other COCOMO drivers when costing the development of a secure system. Our current study makes the contribution of evaluating the common software cost estimations models like FPA, MkII, COCOMO II, SLIM and COSMIC-FFP from the security aspects.

2.2 IFPUG 4.1

In Albrecht Function Point Analysis (FPA) (Albrecht, 1979; Longstreet, 2008, Longstreet Consulting Inc, 2008), there are two parts in the measurement, which are Unadjusted Function Point and Adjusted Function Point. First part consists of five components, which are External Inputs (EI), External Outputs (EO), External Inquires (EQ), Internal Logical Files (ILF) and External Interface Files (EIF), and these components are evaluated by complexity weights. The second part is 14 General System Characteristics (GSCs) that measured on a six point scale. The GSCs are presented in Table 1. Based on the manual (Longstreet, 2008), the security factors are not highlighted in FPA. Existing FPA lacks in emphasizing on security costing in developing application. Besides, the issue of security is discussed by the users. The discussions include the sample scenario, user request, technical design, scenario resolution, and general discussion about user security file, access profile file, external inputs, external outputs, external enquiries and contributions to measurement (Total Metrics, 2001).

From Table 1, there are no security factors or risk assessments listed in the first level of the GSCs. However, the security factors are slackly considered in the second level of GSCs (as shown in Table 2). Therefore, any software that contains security coding or schemes, the existing FPA will not be able to estimate the cost that invoked by the built-in security coding in the software. This added cost is important because highly-secure software will increase costs based on the different models.

2.3 Software Life Cycle Model (SLIM)

Software Life Cycle Model (SLIM) is developed by Putnam (Putnam, 1978). It used validated data from over 2600 projects from industry, which stratified into nine application categories raging from microcode to business systems from IBM Canada Ltd. Laboratory. There are three inputs in SLIM, consists of software size, process productivity parameters, and management constraints. Software size is calculated by using Source Line of Code (SLOC), Function Points, Modules and Uncertainty. Process Productivity consists of methods, skills, complexity and tools. Management Constraints are maximum people, budget, schedule and required reliability.

These three inputs go through three main processes to get the minimum possible schedule, evaluate practical alternatives and optimum estimate to meet constraints. As the outputs for SLIM, there are four graphs namely Staff v.s Time, Cumulative Cost vs. Time, Probability of Success vs Time and Reliability vs. Time. This concept supports the SLIM tool for two important management indicators, which are Productivity Index and Manpower Buildup Index (Panlilio, 1994). Therefore, in Putnam's SLIM, there is no emphasis on the security aspects throughout the system development.

2.4 COSMIC-FFP

The new release COSMIC-FPP (COSMIC, 2007) has changed this method's name to COSMIC Method. COSMIC provides a standardized method of measuring a functional size of software from the functional domains commonly referred to as business application (or MIS) software and real-time software.

The COSMIC measurement method involves applying a set of models, principles, rules and processes to the Functional

User Requirements (or 'FUR') of a given piece of software. The general measurement process consists of three phases, which are the Measurement Strategy Phase, the Mapping Phase, and the Measurement Phase. During the Measurement Strategy Phase, four elements are identified, which are purpose, scope, functional users and level of granularity of the software. The Mapping Phase identifies functional processes, data groups (Entry, Exit, Write and Read data) and data attributes. The last phase is the Measurement Phase. The data movements of each functional process are identified and measured by 1 Cosmic Function Point as the size of one data movement. Then, aggregate the measurement results to get the size of the software. (COSMIC, 2007). Throughout the measurement, the security aspect is slightly mentioned as one of the examples in the Environment Constraints in Functional User Requirements (FUR). Therefore, if a software contains security coding or schemes, the existing COSMIC will not be able to estimate the cost that invoked by the built-in security coding in the software.

2.5 COCOMO II

COCOMO (COnstructive COst MOdel) which created by Barry Boehm is currently established to COCOMO II.2000 (Boehm, 1981; Boehm *et. al*., 2000). The full COCOMO II model includes three stages. Stage 1 supports estimation of prototyping or applications composition efforts. Stage 2 supports estimation in the Early Design stage of a project, when less is known about the project's cost drivers. Stage 3 supports estimation in the Post-Architecture stage of a project.

From the Effort Estimation Equation, COCOMO II (Boehm, 2000) considers the Effort Multipliers, size by using source line of code and adjusted function points, scale factors, and adaptation adjustment factors. Effort Multipliers includes product attributes, platform attributes, personnel attributes, and project attributes. These attributes are as follows:

Product Attributes

- Required software reliability (RELY)
- Product complexity (CPLX)
- Database size (DATA)
- Required Reusability (RUSE)
- Documentation match to life-cycle needs (DOCU)

Platform Attributes

- Execution time constraint (TIME)
- Platform volatility (PVOL)
- Main storage constraint (STOR)

Personnel Attributes

- Analyst capabilities (ACAP)
- Platform experience (PLEX)
- Applications experience (APEX)
- Programming language experience (LTEX)
- Programmer capabilities (PCAP)
- Personnel Continuity (PCON)

Project Attributes

- Use of software tools (TOOL)
- Multisite Development (SITE)

In additional to the 16 adjustment factors, there are two user defined factors, which their initial values are set to 1. In this version of COCOMO, the security aspects are not listed in the calculation. However, Expert COCOMO (Boehm et. al., 2000) aids in project planning by identifying, categorizing, quantifying, and prioritizing project risks. It used an automated heuristic method for conducting software project risk assessment.

2.6 Mk II FPA

Mk II Function Point Analysis (Mk II FPA) (Symons, 1991; UKSMA, 1998) is one of the most widely used SCE that managed by United Kingdom Software Metrics Association (UKSMA). It is a method for the quantitative analysis and measurement of information processing applications. It quantifies the information processing requirements specified by the user to provide a figure that expresses a size of the resulting software product. This size is suitable for the purposes of performance measurement and estimating in relation to the activity associated with the software product (UKSMA, 1998). Mk II FPA measures the functional size based on logical transactions concept. Each logical transaction consists of three components; input across an application boundary, processing involving stored data within the boundary and output back across the boundary. Besides, MK II measures the influence on the size of the application of each of 19 (or more user defined characteristics) technical characteristics on a scale of 0 to 5. Refer to Table 3 for the characteristics (UKSMA, 1998). The sum for all characteristics or Total Degrees of Influence (TDI) is used to compute Technical Complexity Adjustment (TCA). Function Point Index (FPI):

$FPI = Wi * \sum Ni + We * \sum Ne + Wo * \sum No$

where

$Wi = 0.58$; $We = 1.66$; $Wo = 0.26$; Ni = Input Data element Types;

Ne = Data Entity Types Referenced; No = Output Data Element Types;

Technical Complexity Adjustment (TCA):

$TCA = (TDI * C) + 0.65$

where

Current Industry Average Value of $C = 0.005$;

Adjusted Function Point Index (AFPI):

$AFPI = FPI * TCA$

From Table 3, there is a characteristic related to Security, Privacy, Auditability to measure the level of confidentiality or security. But the related scores and descriptions are as in Table 4. From Table 4, the scores for security, privacy and auditability are quite low. Besides, the details of risk assessment or security are not listed here. Therefore, when a software is embedded with security coding, the existing Mk II might under estimate the cost that invoked by the built-in security coding in the software.

3. Security standards

There are four security standards that are reviewed in this paper, which are commonly used as guidelines in considering the security cost in system and web development.

3.1 IT Security Cost Estimation Guide

This Information Technology Security Cost Estimation Guide (Department of Education, 2002) provides the security cost estimation based on Management Controls, Operational Controls and Technical Controls. For each control, there are categories of potential security risks that a developer has to estimate while calculating for the security cost. To effectively use this guidance, the user has to complete the National Institute of Standards and Technology (NIST) Self Assessment Guide for Information Technology Systems. According to this guide, security must be considered in integral part of the overall IT infrastructure in order to effectively manage risks. As risks vary by initiative and fluctuate throughout the life cycle of each system, the necessary costs for security controls will vary as well. Therefore, risks should be identified according to the life cycle phases during which they will pose the most imminent threat, as this will largely determine which fiscal year to budget for controls (Department of Education, 2004).

This guide suggested an IT Security Costing Framework. This framework comply with mandated standards of information system security, and each system, at a minimum, must meet requirements that fall within the following 17 management, operational and technical control categories. Refer to Table 5. These 17 categories align with the 17 elements of IT security identified in the NIST Self Assessment. This guide provides the calculation of costs based on government employees or contractors rates.

3.2 Common Criteria for Information Technology Security Evaluation

Common Criteria for Information Technology Security Evaluation (CC v 3.1) (CC, 2008) consists of three main parts: Part 1 – Introduction and general model; Part 2 – Security functional requirements; Part 3 – Security assurance requirements. This guide is used by COCOMO II to extend its model to COSECMO (Colbert et. al., 2006; Wu et. al., 2006). This CC philosophy is to provide assurance based upon an evaluation of the IT product or system that is to be trusted. The assurance classes include configuration management, delivery and operation, development, guidance documents, life cycle support, tests, vulnerability assessment, protection profile and security target evaluation.

Assurance family from CC v 3.1 lists the following assurance classes to be considered during the security evaluation:

- Configuration Management (ACM)
- Delivery and Operation (ADO)
- Development (ADV)
- Guidance documents (AGD)
- Life Cycle Support (ALC)
- Tests (ATE)
- Vulnerability Assessment (AVA)
- Protection Profile Evaluation (APE)
- Security Target Evaluation (AST)

For each assurance family, brief description is provided to help the evaluators to assess the product or system. The evaluation assurance level is from scale EAL1 to EAL7. For each scale, there are some assurance components to be evaluated (CC, 2008).

3.3 The Open Web Application Security Project (OWASP)

The Open Web Application Security Project (OWASP) (OWASP, 2008) is a worldwide free and open community focused on improving the security of application software. It is an open community dedicated to find and fight the causes of insecure software. It is a guide for building secure web applications and web services. OWASP has chosen Microsoft's threat risk modeling process as it works well for the unique challenges facing application security. OWASP guidelines list the following aspects to be considered:

- Threat Risk Modeling
- Handling e-Commerce Payments
- Phishing
- Secure Web Services
- Authentication
- Authorization
- Session Management
- Data Validation
- Interpreter Injection
- Canoncalization, Locale and Unicode
- Error Handling, Auditing and Logging
- File System Protection
- Buffer Overflows
- Administrative Interfaces
- Cryptography
- Configuration
- Maintenance
- Denial of Service Attacks

OWASP guidelines help to encounter the causes of the insecure software. It lists the risk factors that potentially attack an application. This security factors should be considered during the application development. Furthermore, these factors should evolve some costs in developing secure software (OWASP, 2008).

3.4 Control objectives for information and related technology (COBIT)

Control Objectives for Information and related Technology (COBIT) (ISACA, 2008) provides good practices across a domain and process framework and presents activities in a manageable and logical structure. COBIT's good practices represent the consensus of experts. These practices will help optimize IT-enabled investments. One of the objectives of COBIT is to ensure IT risks are managed appropriately. Therefore, the focus area includes risk management, which requires risk awareness. COBIT is a process-oriented and controls-based framework. There are five IT Governance focus areas, which are namely strategic alignment, value delivery, risk management, resource management and performance management. It is organized according to IT process like plan and organizes, acquire and implement, deliver and support, monitor and evaluate. During plan and organize, the user of this guide has to assess and manage IT Risks (P09). In the process of Acquire and Implement, the user has to Acquire and Maintain Technology Infrastructure (AI3). The most important process that related to security is Deliver and Support, which includes Ensure Continuous Service (DS4) and Ensure Systems Security (DS5). The aspects and considerations of COBIT are as follows:

- PO2 Define the Information Architecture
- PO3 Determine Technological Direction
- PO9 Assess and Manage IT Risks
- AI2 Acquire and Maintain Application Software
- AI3 Acquire and Maintain Technology Infrastructure
- DS4 Ensure Continuous Service
- DS5 Ensure Systems Security
- DS11 Manage Data
- DS12 Manage the Physical Environment
- ME2 Monitor and Evaluate Internal Control
- DS13 Manage Operations

4. Methodology

This methodology produces the proposed potential security factors. It consists of:

a. Review on the existing software cost estimation models such as Mk II, COSMIC-FFP, COCOMO II, SLIM and FPA, concentrating on the security aspect.

b. Review four security standards that related to IT and development like IT Security Guidelines, CC v 3.1, COBIT 4.1 and OWASP, for identifying the potential security factors that can be included in software cost estimation.

c. Integrate the lists from four security standards and categories the items into lists according to a proposed potential security factors.

Matching the proposed potential security factors in System Development Life Cycle (SDLC), which include Plan, Design, Code, Test and Deploy.

5. Results and discussions

The integrated list of proposed potential security factors is combined from four security standards and organized according to the software development life cycle as shown in Table 6. All these lists are integrated according to the steps and proposed potential security factors in Table 6. This integrated list forms a framework that helps the practitioners and developers to consider the potential security risks while planning for an application (Refer to Figure 1).

List for Security Requirements (SR)

- Specification of Management
- Risk Analysis
- PHP Guidelines
- Abuse Cases
- Management of IT Security
- Functions
- Threat Risk Modeling
- Security Rules
- IT Security Plan

List for Security Features (SF)

- User Identify
- Authentication
- Management
- Session Management
- Interpreter Injection
- Cryptography
- Authorization
- Exchange of Sensitive Data
- User Attribute Definition
- Verification of Secrets
- Protection of Security
- Data Integrity
- Data Validation

List for Functional Features (FF)

- Management Security
- Handling E-Commerce Payments
- Production and Input/Output Controls
- Configuration
- File System
- Functions Behavior
- Web Services
- Personnel Security
- Incident Response Capability

List for Attack Planning (AP)

- Phishing
- Malicious Software Prevention, Detection and Correction
- Denial of Service Attacks
- Non-bypass Ability of TSP

List for Formal Review & Sign Off (FR)

- Review of Security Controls
- Audit Data Generation
- Logical Access Controls
- Security Attribute based Access Control

List for Software Security Assurance (SSA)

- Security Testing
- Security Surveillance
- Security Monitoring

List for Final Security Review & Sign Off (FSR)

- Final Review of Security Controls
- Final Logical Access Controls
- Final Security Attribute based Access Control
- Final Audit Data Generation

List for Infrastructure Application Security Measures (ASM)

- Security Awareness
- Security Training
- Security Education

List for Software Hardening & Application Security Monitoring (SHA)

- Hardware Maintenance
- System Software Maintenance

These potential security factors are able to help the practitioners and developers to estimate the software security costs.

The proposed potential security factors are the factors that might influence the software cost estimation. As mentioned before, security is often an afterthought when developing software and is often bolted on late in development or even during deployment or maintenance. Any occurrences that caused by the security aspects will add to the existing software cost when the developers fix the problems. It is better to discover and estimates the security cost before or during the development of application. Therefore, this proposed potential security factors is a big help to the practitioners or developers while designing or developing any application.

6. Conclusions

Software houses are now keen to provide secure software as requested by customers' desire with respect to security and quality of their products (Jari, 2006). In examining some real-world experiences, project managers faced project "fires" when risks were not identified during the planning phase. These fires often resulted in added costs or negative consequences (Frederick, 2006). Engineering security will substantially raise software – project cost and there has been wide variation in the amount of added cost estimated by different models (Colbert et. al., 2006). Therefore, there is a need to include secure software costing while estimating the application cost. Based on the review of current software cost estimation models and security standards, a proposed potential security factors is suggested. This integrated list can help the software developers to encounter the built-in security factors in the application by estimating the application cost more precisely.

In the future, extending the existing software cost estimation technique is essential in order to calculate the cost that includes security in application. We also decide to visualize the integrated list by developing a prototype of software cost estimation tool and validate it through case studies in the system development companies.

References

Albrecht, A.J. (1979). Measuring application development productivity. In: SHARE/GUIDE: Proceedings of the IBM Applications Development Symposium, 79:(83-92), 1979.

Boehm, B. (1981). Software Engineering Economics. Englewood CI@, NJ Prentice-Hall, 1981.

Boehm, B., Abts, C., Brown, A.W., Chulani, S., Clark, B.K., Horowitz, E., Modachy, R., Reifer, D., Steece, B. (2000). Software Cost Estimation with COCOMO II. Prentice Hall, Inc., New Jersey. ISBN 0-13-026692-2, 2000.

Briand, L.C. El Emam K., Maxwell, K., Surmann, D., Wieczorek, I. (1998). An Assessment and Comparison of Common Software Cost Estimation Models. Technical Report, ISERN TR-98-27, International Software Engineering Research Network.

Briand, L.C., El Emam, K., Surmann, D., Wieczorek, I., Maxwell, K.D. (1999). An Assessment and Comparison of Common Software Cost Estimation Modeling Techniques. ICSE '99:(313-322), Los Angeles CA, ACM 1999.

CC. (2008). Common Criteria for Information Technology Security Evaluation, version 3.1. [Online] Available: http://www.commoncriteriaportal.org.

Chulani, S., Boehm, B., Abts, C. (1998). Software Development Cost Estimation Approaches – A Survey. University of Southern California – Computer Science Engineering, 1998.

Colbert E., Wu D., Chen Y., and Boehm B. (2004). Cost Estimation for Secure Software and Systems. University of Southern California, United States of America.

Colbert E., Wu D., Chen Y., and Boehm B. (2006). Cost Estimation for Secure Software and Systems. University of Southern California, USA.

Common Software Measurement International Consortium (COSMIC) (2007). The COSMIC Functional Size Measurement Method Version 3.0 – Measurement Manual. September 2007

Department of Education, USA. (2002). Information Technology Security Cost Estimation Guide. November 2002.

Frederick, L. (2006). The Importance of Identifying Risk during Project Planning. In Ackermann, D., Project Management In Practice, Boston University, Metropolitan College, The 2006 Project Risk and Cost Management Conference, Commonwealth Avenue.

ISACA. (2008). Control Objectives for Information and related Technology (COBIT) – Framework, Control Objectives, Management Guidelines, and Maturity Models. IT Governance Institute, USA.

Jari, R, (2006). Contracting over the Quality Aspect of Security in Software Product Markets, University of Lapland, Rovaniemi, Finland. ACM Press, Oct 2006.

Kemerer, C.F. (1987). An empirical validation of software cost estimation models. Communications of the ACM vol. 30, no. 5, 416-429, May 1987.

Longstreet Consulting Inc. (2008). [Online} Available: http://www.SoftwareMetrics.com

Longstreet, D. (2008). Function Points Training and Analysis Manual. ISBN: 0-9702439-3-6, Jan 2008.

OWASP. (2008). The Open Web Application Security Project – A Guide to Building Secure Web and Web Services. Black Hat 2nd Ed.

Panlilio N.Y. (1994). Software Estimation Using the SLIM Tool. IBM Canada Ltd Laboratory Technical Report TR 74.102.

Putnam, L.H. (1978). A general empirical solution to the macro software sizing and estimation problem. IEEE Transactions on Sofiware Engineering, ~01.4, no. 4, 345-381, July 1978.

Shanai Ardi, David Byers, Nahid Shahmehri (2006). Towards a Structured Unified Process for Software Security, ACM Press, May 2006.

Srinivasan, K., Fisher, D. (1995). Machine learning approaches to estimating software development effort. IEEE Transactions on Software Engineering, Vol. 21, no. 2, 126-137, February 1995.

Symons, C. (1991). Software Sizing and Estimating – Mark II FPA. Wiley Series in Software Engineering. John Wiley and Sons, U.K., ISBN 0-471-92985-9, 1991.

Total Metrics. (2001). [Online] Available: http://www.totalmetrics.com

United Kingdom Software Metrics Association (UKSMA). (1998). Mk II Function Point Analysis – Counting Practices Manual Version 1.3.1. September 1998.

Wu, D., and Yang, Y. (2006). Towards An Approach for Security Risk Analysis in COTS Based Development. Center for Software Engineering, University of Southern California, Los Angeles, CA. Proceeding of Software Process Workshop on Software Process Simulation.

Table 1. List of General System Characteristics (GSCs)

Characteristics	Brief Descriptions
Data communications	How many communication facilities are there to aid in the transfer or exchange of information with the application or system?
Distributed data processing	How are distributed data and processing functions handled?
Performance	Did the user require response time or throughput?
Heavily used configuration	How heavily used is the current hardware platform where the application will be executed?
Transaction rate	How frequently are transactions executed daily, weekly, monthly, etc.?
On-Line data entry	What percentage of the information is entered On-Line?
End-user efficiency	Was the application designed for end-user efficiency?
On-Line update	How many ILF's are updated by On-Line transaction?
Complex processing	Does the application have extensive logical or mathematical processing?
Reusability	Was the application developed to meet one or many user's needs?
Installation ease	How difficult is conversion and installation?
Operational ease	How effective and/or automated are start-up, back up, and recovery procedures?
Multiple sites	Was the application specifically designed, developed, and supported to be installed at multiple sites for multiple organizations?
Facilitate change	Was the application specifically designed, developed, and supported to facilitate change?

This list of 14 General System Characteristics (GSCs) is being measured as part of the Adjusted Function Point Analysis. The characteristics are measured on a six point scale, which is from 0 to 5.

Table 2. Security features and descriptions in the second level of GSCs

Feature	Characteristic	Descriptions / Second Level
Security	Heavily Used Configuration	Some security or timing considerations are included.
	Complex Processing	Sensitive control (for example, special audit processing) and / or application specific security processing

This table shows the characteristics and descriptions that related to security factors. There are only two GSCs that have concerned in security factors in their second level. Heavily Used Configuration has a score of 2 for its second level while Complex Processing has a score of 3.

Table 3. Technical Characteristics

Characteristics	Brief Descriptions
Data communication	Data and control information use communication facilities
Distributed Function	Application spread over two or more processors
Performance	Application response/throughout influence the application
Heavily used configuration	The equipment that will run the application is already heavily used
Transaction rates	The high rate of arrival of transactions causes problems beyond those of Performance
On-Line data entry	Terminal used for input
Design for End-user efficiency	Human factors considered in design
On-Line update	Data updated in real time
Complexity of processing	Internal complexity beyond that dealt with by entity counting conventions
Usable in Other Applications	The code is designed to be shared with or used by other applications
Installation ease	Data conversion and ease of installation were considered in design
Operations ease	Ease of operation was considered in design
Multiple sites	The application is to be used in many sites and/or many organizations
Facilitate change	Future changes to the application a consideration on design
Requirements of Other Applications	Interfaces
Security, Privacy, Audit ability	Special features of confidentiality / security
User Training Needs	Specific requirements
Direct Use by Third Parties	Degree of use/connection to the application
Documentation	-

From Table 3, there are 19 technical characteristics on a scale of 0 to 5. These characteristics measure the influence on the size of the application.

Table 4. Second level of Technical Characteristics

Score	Brief Descriptions
1	If the application has to meet personal, possibly legal, privacy requirements
1	If the application has to meet special auditability requirements
2	If the application has to meet exceptional security requirements
1	If encryption of data communications is required

Table 4 shows the second level of Technical Characteristics for Mk II. As noticed, the scores for security, privacy and auditability are ranged from 1 to 2. There are no details of risk assessment or security listed here.

Table 5. IT Security Costing Framework

Type of Control	Categories
Management	Risk Management
	Review of Security Controls
	Life Cycle
	Authorize Processing (Certification & Accreditation)
	System Security Plan
Operational	Personnel Security
	Physical and Environment Protection
	Production and Input/Output Controls
	Contingency Planning
	Hardware and System Software Maintenance
	Data Integrity
	Documentation
	Security Awareness, Training, and Education
	Incident Response Capability
Technical	Identification and Authentication
	Logical Access Controls
	Audit Trails

Table 5 shows the requirements that fall within the 17 management, operational and technical control categories, which align with the 17 elements of IT security identified in the NIST Self Assessment.

Table 6. Proposed Potential Security Factors in System Development Life Cycle

Step	Proposed Potential Security Factors
Plan (P)	Security Requirements (SR)
Design (D)	Security Features (SF)
	Functional Features (FF)
Code (C)	Attack Planning (AP)
	Formal Review & Sign Off (FR)
	Security Coding, Review & Audit (SCR)
Test (T)	Software Security Assurance (SSA)
	Final Security Review & Sign Off (FSR)
	Infrastructure Application Security Measures (ASM)
Deploy (E)	Software Hardening & Application Security Monitoring (SHA)

Table 6. shows an integrated list of proposed potential security factors, which is extracted from four security standards. These potential security factors are organized according to the software development life cycle, which consists of a cycle of Plan, Design, Code, Test and Deploy.

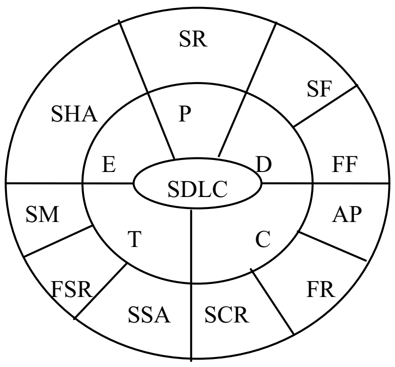

Figure 1. Proposed Potential Security Factors Framework

This framework is based on phases of SDLC and an integrated list that extracted from four security standards.