2007

Specifying security requirements improvement for IEEE Standard 830

Jacob D. McCarty
West Virginia University

SPECIFYING SECURITY REQUIREMENTS IMPROVEMENT FOR IEEE STANDARD 830

Jacob D. McCarty

Thesis submitted to the College of Engineering and Mineral Resources at West Virginia University in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science

Roy S. Nutter, Ph.D., Chair
John M. Atkins, Ph.D.
Cynthia D. Tanner, M.S.C.S.

Lane Department of Computer Science
and Electrical Engineering

Morgantown, West Virginia
2007

Keywords: IEEE std. 830, software engineering, SRS, software requirements specifications, security, security legislation.

ABSTRACT

SPECIFYING SECURITY REQUIREMENTS IMPROVEMENT FOR IEEE STANDARD 830

Jacob D. McCarty

This paper presents a concept on how the software requirements specifications template provided by IEEE Standard 830 could be updated to ensure that security is analyzed during the early stages of the software development lifecycle. This improved security requirement in the software requirements specifications will ensure that software developers will have a more clear understanding of how to protect digital information.

DEDICATED TO THE MEMORY OF
John Robert “Bob” Moran and Emma Teresa Moran,
their knowledge and inspiration
will forever be passed on from
generation to generation.

ACKNOWLEDGEMENTS

I would like to thank the following people for their help:

Dr. Roy Nutter, for agreeing to be my chair and guiding me through the thesis process,

Dr. John Atkins, for enlightening me where I should have been enlightened,

Cindy Tanner, for providing me wisdom in my time of need,

Linda Kress and **Tammy McDonald**, for keeping me sane during my graduate studies years.

I would also like to thank the senior design team that helped me redesign BTM, which was the inspiration for this thesis:

Sarah Lovell
Nick Bialaszewski
Shawn Holstein

Last but not least, I would like to thank **my family** for encouraging me to get the most out of my education.

TABLE OF CONTENTS

TABLE OF FIGURES

1 INTRODUCTION

Professionals in the field of software engineering have taken many steps to enhance the design of software to ensure secure information. Security has traditionally been an afterthought of the computer software design process [16]. Current standards recommend that security be evaluated in the software requirements specifications. "It is not that developers are incapable of producing [secure] software … it is just that they are not sufficiently motivated to do so" [15]. Developers generally do not understand the security requirements of software systems that they are designing; therefore, such security requirements are either ignored or not adequately fulfilled.

This paper presents a discussion of current software practices of developing the software requirements specifications. This document begins by presenting a brief background of software engineering processes and the phases in which software requirements specifications are developed. Next it presents the Institute of Electrical and Electronics Engineers, IEEE, standard for software requirements specifications. Third it presents examples of current legislation and regulations surrounding the use of sensitive information. Finally, a possible solution on how to further define security in the software requirements specifications is presented.

1.1 Statement of the Problem

Current practices for developing software requirements specifications appear to be inadequate. The current software requirements specifications standard does not provide a clear description on how to specify security requirements. This document provides a detailed method for improving the IEEE standard regarding security. The present IEEE Standard 830 places security information in the software attributes section and does not provide a clear description of how to specify security requirements. This thesis will provide a suggested outline for developing software requirements specifications with improved security visibility.

2 BACKGROUND

Software engineering can be described as a process in which a computer program and its supporting documentation are developed. There are many types of software process models that are used to manage how a software product is developed. According to Ian Sommerville, "Most software process models are based on one of three general models or paradigms of software development" [11]. The three general models are: the waterfall approach, iterative development, and component-based.

2.1 The Software Process

All of the software models produce many different types of documentation to describe the software being developed. These documentation sets serve as contracts between the users of the system, the client asking for the product being developed, and the software development team. After development is completed, testing of the software begins, based on the development documentation, to ensure that all aspects of the software were developed to the software requirements specifications. This paper will describe the IEEE standard to developing software requirements specifications and make recommendations to update the standard to meet the needs of organizations developing software. This paper will explain the different models and techniques to developing software and point out the specific location in which software requirements specifications are created.

In 1968 and 1969, software engineering became the official practice for developing computer software architectures and designs during two NATO Software Engineering Conferences [10]. During these conferences, software engineering was compared to computer science and practitioners of both disciplines discussed how they could work together to develop better software products. The discussion was completed by a group comprised of academic and industry professionals. These conferences set the concept of computer software development being a set of phases: conception, design, implementation, testing, and maintenance [10]. Software models take a different approach to completing these phases, but every model discussed hereafter contains the concepts of the phases put forward at the conference.

In the forthcoming sections there is a brief description of three different software lifecycle models. Note that in each of the descriptions there is a specific notation stating which phase or phases the software requirements specifications are developed.

2.1.1 The Waterfall Model

The waterfall model, created by W. W. Royce in 1970, is a set of incremental steps. Each step is considered a phase in which its predecessor must be completed prior to moving forward in the development process. This lifecycle model includes the following phases, listed in order: requirement definition, system and software design, implementation and unit testing, integration and system testing, and operation and maintenance. The waterfall model provides a means by which developers can reevaluate a previous phase. If a problem is discovered, the development team suspends the current phase and reenters the previous phase to correct problems prior to moving forward with development. Due to the specific set of phases and how they are to be completed, the waterfall model is typically not a good model for use in software design where the system requirements are not well understood or are expected to rapidly change throughout the process. During the development of the waterfall model, security was not an issue that needed to be highlighted; therefore, it was left out of the model for analysis. The software requirements specifications is completed in the second phase of the waterfall model [11]. Figure 1 displays a graphical representation of the waterfall model.

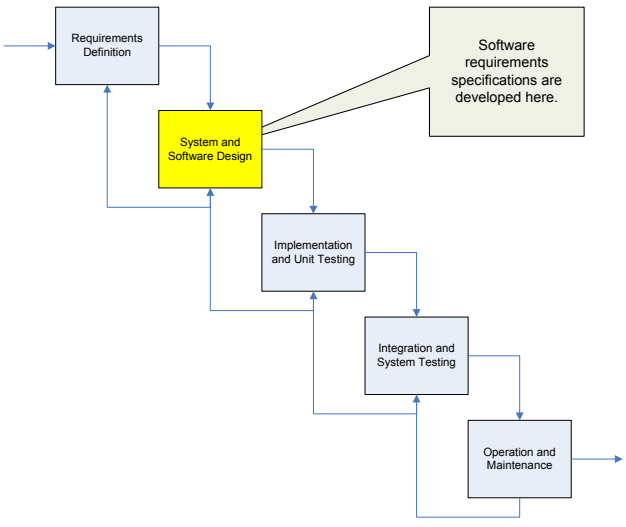

Figure 1: The Waterfall Model [11]

2.1.2 The Spiral Model

The spiral model, created by B. W. Boehm in 1988, is represented as a series of spirals. The software process begins in the innermost spirals and work outward. Each iteration of the spiral focuses on a different aspect of the software being developed. The spiral model analyzes each cycle in four ways: objective setting, risk assessment and reduction, development and validation, and planning. The main focus of the spiral model is risk; if risk is determined to be too high, then the project is ended and not completed. During the risk assessment a security analysis should be completed, if the security risk is too high then the project would be ended. The software requirements specifications are completed in one cycle of the spiral [11]. Figure 2 provides a graphical representation of the spiral model.

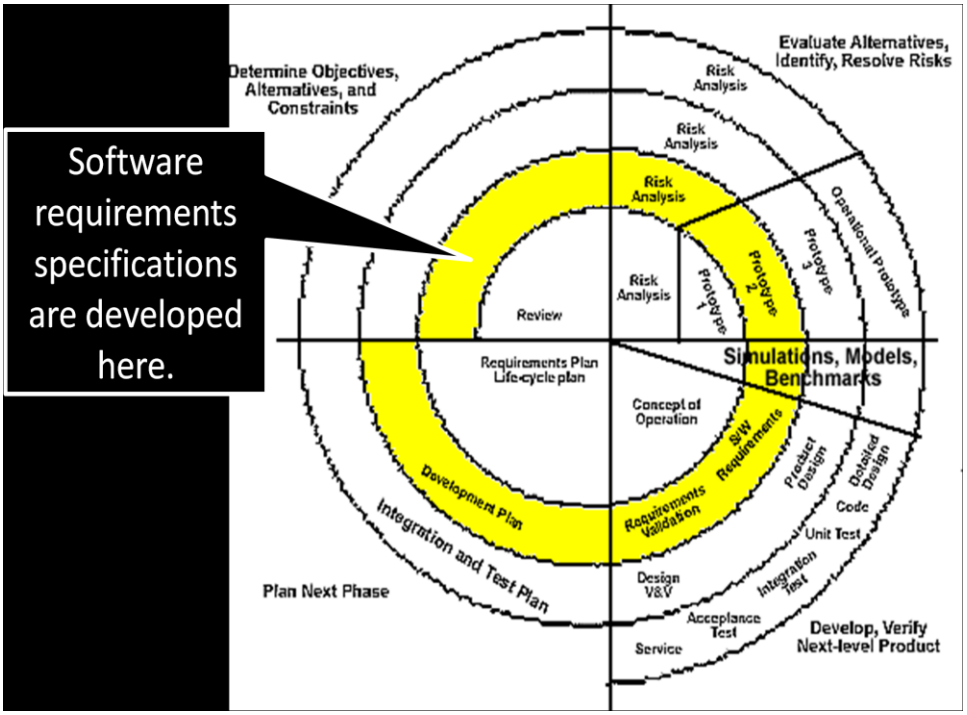

Figure 2: The Spiral Model [11]

2.1.3 The Unified Software Development Process

The unified process, created by J. Rumbaugh, I. Jacobson, and G. Booch in 1999, is divided into four areas of focus: inception, elaboration, construction, and transition. The unified process is considered an iterative process. Each of the specific phases is reviewed in an iterative fashion followed by the complete process being iterated for the next component in the system. Each iteration of the process focuses on a specific module in the complete system based on the ranked business needs. This model focuses on business concerns rather than technical concerns. The software requirements specifications are completed throughout all phases, but the majority of the specifications are developed during the inception and elaboration phases. The unique factor in the unified process is that it focuses on what it considers "six fundamental best practices." These fundamentals are: develop software iteratively, manage requirements, develop user component-based architectures, visually model software, verify software quality, and control changes to software [11]. During development with the unified process model, security requirements would be gathered as a step within each iteration of a phase. During the inception phase security would be specified as an overview. During the elaboration phase, security would be specified in software requirements specifications. The construction phase would focus on how to code securely, and in the transition phase physical security measures would be placed into the system. Figure 3 provides a graphical representation of the unified process model.

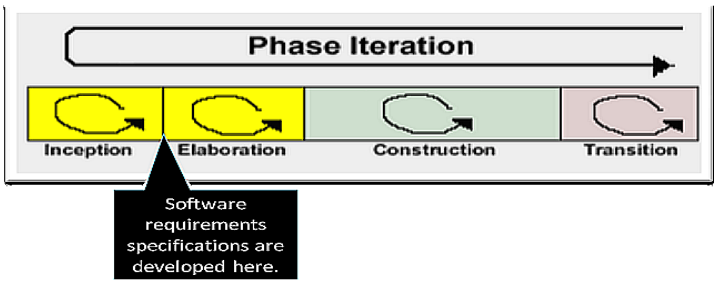

Figure 3: The Unified Software Development Process [11]

2.2 IEEE Standard 830

The Institute of Electrical and Electronic Engineers, IEEE, is a nonprofit professional association that strives to advance technology. It is comprised of industry professionals, academic professionals, and students. IEEE has produced many standards for engineering by using a set of tested and scrutinized methods. "Our standards are developed in a unique environment that builds consensus in an open process based on input from all interested parties" [7]. IEEE believes that by providing and defining standards for the technology industry, organizations will have the following benefits:

> "...market growth for new and emerging technologies, reduced development time and cost, sound engineering practices, decreased trading costs and lowered trade barriers, increased product quality and safety, reduced market risks, and protection against obsolescence" [7].

There are currently three publications of the IEEE Standard 830: Release 1984, Release 1993, and Release 1998. These standards describe what a high-quality software requirements specifications document should contain and how it should be organized. The only main difference between the three documents is how the information is visually presented in each release, but the concepts and templates are essentially the same. IEEE states that all of their standards must be reviewed every five years.

2.2.1 Software Requirements Specifications Qualities

The software requirements specifications should be an unambiguous, verifiable base for an agreement between the customer and the developer as to what is to be designed. This understanding should be based on the following characteristics of good software requirements specifications: "correct, unambiguous, complete, consistent, ranked for importance and/or stability, verifiable, modifiable, and traceable" [13]. The main goal of the document is to reduce the cost – both time and financial – of the

development process. During the later phases of the project, the software requirements specifications is used to validate and verify that all contractual agreements have been achieved by the development team, and also serves as a reference to individuals or organizations performing maintenance on the software product after it has been delivered to the customer [2].

IEEE Standard 830 provides templates for the software requirements specifications to the industry. A sample software requirements specifications template is provided in section 6.1. This template shows many aspects that are needed to properly specify requirements for a software project; a description of the sections of the template is provided in the following sections.

2.2.1.1 SRS: Introduction Section

The introduction section is designed to provide information to the user that might be helpful while progressing through the software requirements specifications document. The purpose section is to specify the reason for the software requirements specifications as well as the target audience. The scope section provides the names of the software products to be designed and the main goals and objectives of the system. If there is anything specific the software product will not accomplish, this is to be clearly stated here. The definitions, acronyms, and abbreviations section provides a reference area for the reader to refer to while reading the document. This can be a bulleted list or in any format, but should explain any unclear terms or technical aspects that either the customer or the developer might not understand while reading the software requirements specifications. The references section is expected to list any referenced documents during the creation of the software requirements specifications. The overview section should explain what the rest of the software requirements sections either mean or entail [13].

2.2.1.2 SRS: Overall Description Section

The overall description area provides information on how each of the factors of the system affects the software requirements specifications. This section is not to include specific requirements; specific requirements are placed in the specific requirements section of the document. The product perspective section should describe the system in terms of other products, either on the market or currently being used in the old environment – the system being replaced. The product functions section is to briefly describe the major components of the system being developed. Graphical representation may be presented in this section to help the reader understand each function's relationship to other functions and the system as a whole. User characteristics should describe the users' knowledge base. This should not provide requirements the users will need to use the system, but provide a better understanding as to why the system is being designed in a specific manner. The constraints section is to describe what constraints might be put on the system being designed. For instance, if the software being designed is to be used on cellular devices; then, the application will have less memory to operate in comparison to an application being deployed on a desktop. Assumptions and dependencies are listed in the software requirements specifications because most systems do not perform correctly due to developers or users assuming that the other party has a clear understanding of a requirement which might not have been acknowledged by the other party. The items listed in the assumptions and dependencies area are to explain requirements that might affect the software requirements specifications [13]. For example:

> "…an assumption may be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the SRS [software requirements specifications] would then have to change accordingly" [13].

The apportioning of requirements section is used to explain certain features or functionality that might be delayed for future releases or versions of the software.

2.2.1.3 SRS: Specific Requirements

The specific requirements area is to specify the software requirements in a clear and technical manner so the developers can complete development to the needs of the customer. The external interfaces section provides a detailed description of all inputs into the system and outputs returned by the system. This is completed by breaking down all the data inputs/outputs and describing the details about them. The functions section should provide technical details about all of the functions provided in the software. This is completed using both textual and graphical descriptions of the following areas: validity checks on the inputs, exact sequence of operations, responses to abnormal situations, effects of parameters, and relationship of outputs to inputs. The performance requirements section should provide system performance requirements. For example, time expectations for specific operations, the number of terminals to be supported, and the type of information to be handled. The logical database requirements section provides a description of the database, if necessary. The features that the section analyzes are: types of information used by various functions, frequency of use, accessing capabilities, data entities and their relationships, integrity constraints, and data retention requirements. The standard compliance section provides items that constrict the design to specific formats. These typically occur during reporting of information in the system for audit purposes – a specific report for a government organization. The software system attributes section defines the reliability, availability, security, maintainability, and portability of the system. External interface requirements provide information to help the developers and users of the system understand how the new software will interact with other entities in the system's environment. [13].

This paper will discuss in more detail suggested methods on improving the IEEE standard concerning security. The standard places the majority of security information into the software attributes section and does not provide a clear understanding to what security requirements mean or how they should be developed.

2.2.2 SRS: Security Requirements Evolution

Security Requirements in IEEE Standard 830 have not evolved during each release. In all releases, IEEE Standard 830-1984, IEEE Standard 830-1993, and IEEE Standard 830-1998, the security requirements were specified under the subsection of attributes in the specific requirements section. The security specifications area stated that it should address factors such as "accidental or malicious access, use, modification, destruction, or disclosure" [2].

2.3 Legislation and Regulations

New legislation, regulations, and corporate policies affect how information technology is used to secure sensitive information. Legislation is currently being developed throughout the federal and state levels of the United States government to ensure that personal information is not disclosed without the explicit consent of the United States' economic consumers. The forthcoming sections will describe some examples of such regulatory efforts.

2.3.1 United States Federal Legislation

Information security is gaining momentum throughout the United States. Federal legislation is pushing the information technology sector to secure sensitive information. A few examples of these laws follow.

2.3.1.1 Family Educational Rights and Privacy Act

The Family Educational Rights and Privacy Act of 1974, also known as FERPA, protects students' education records. Information that is considered private according to FERPA includes, but is not limited to, academic performance and financial account information. This federal regulation does permit directory information to be released to the public under the guidelines that such information is public knowledge [5].

2.3.1.2 Health Insurance Portability and Accountability Act

The Health Insurance Portability and Accountability Act of 1996, also known as HIPAA, provides regulatory standards on how electronic medical information is to be handled by health care organizations. This statute provides protection against many abuses in the health care industry. It specifically states that if an individual or organization gains unauthorized access to any unique health care identifier, personal identifiable medical information, or discloses such information that the individual or organization is punishable by fine and/or imprisonment [1].

2.3.1.3 Financial Services Modernization Act

The Financial Services Modernization Act of 1999, also known as the Gramm-Leach Bliley Act, was designed to protect consumer financial information. The Gramm-Leach Bliley Act provides a means for enforcement agencies to enforce two regulations: the Financial Privacy Rule and the Safeguards Rule [14]. The Financial Privacy Rule states that financial institutions must inform consumers of the collection of personal financial information, with whom it will be shared, and how the financial information is going to be protected. This rule also provides a means by which consumers can object to their information being shared with third parties [6]. The Safeguards Rule clearly states that organizations that collect financial information must take measures to protect the information they are provided during transmission and storage [14].

2.3.1.4 Public Company Account Reform and Investor Protection Act

The Public Company Account Reform and Investor Protection Act of 2002, also known as Sarbanes-Oxley, sets forth a few parameters that are pertinent to software design. One of the parameters requires financial audit information to be kept securely for a period of five years. Another parameter states that any mutilation or altering of information is punishable by fine and/or imprisonment. One other parameter that can directly affect how software is designed states that all communications, physical or electronic, must be stored if it pertains to an audit/review or financial information that would/could be audited [12].

2.3.2 United States State Legislation

There is currently Security Breach Legislation in more than half of the United States. These legislative laws are not the only state laws that can affect software engineering, but they provide a clear example how state law can affect the design of software systems. Figure 4 provides a visual understanding of the states with current security breach legislation and the year their legislation went into effect.

The state laws regulating personal information are designed to force industry to take measures to prevent personal information from being stolen or disclosed to unauthorized individuals. The laws state that personal information is, but not limited to: social security number, driver's license, credit card number, debit card number, financial account number, passwords, personal identification numbers, security codes, access codes, and et cetera [3]. All of the current legislation specifically states that if the information disclosed was unencrypted that the individuals of said states must be notified that their personal information may have been disclosed without consent [9].

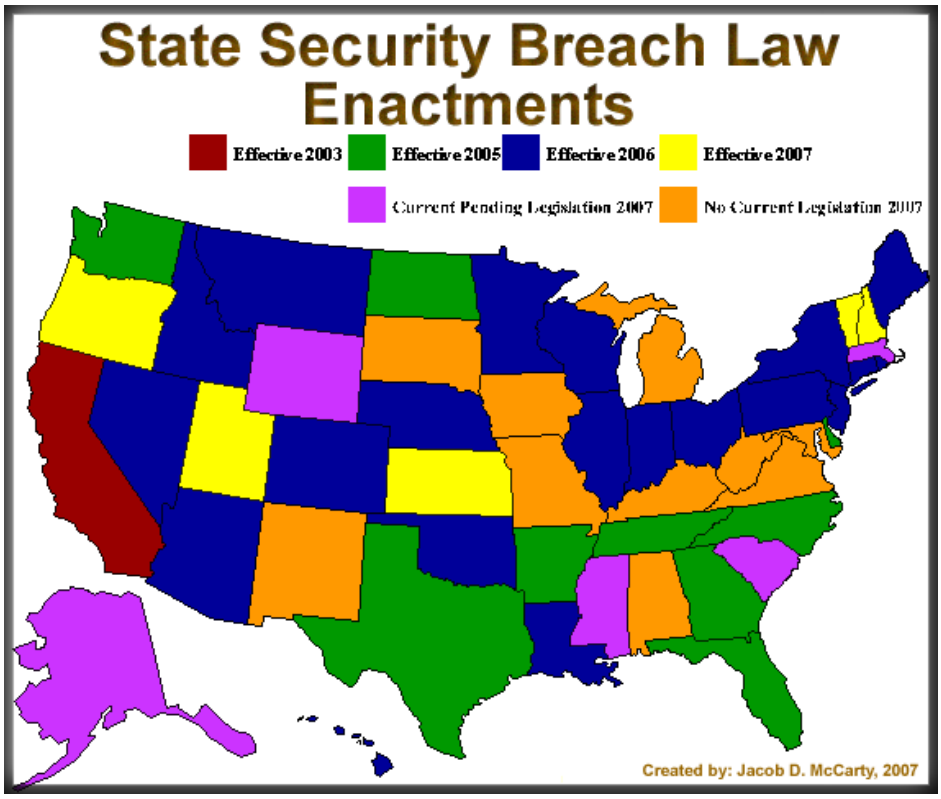

Figure 4: Security Breach Law Enactments [9]

2.3.3 Corporate Policies and Standards

Role-based access control policies are typically seen in corporate regulations. Most organizations set a specific type of role for each of its users. This role based access control policy provides specific credentials to be met prior to permitting a user access to the digital information requested. Information that corporate organizations store, manipulate, and transmit is accepted as needing to be classified and secured.

2.3.3.1 Information Management Security Policy

One of the first steps to creating an information management security policy is specifying the organization's assets. These assets range from employees to digital information. The next step is defining how to protect the organization's assets. A closer look at digital information is needed.

Digital information is typically stored in data centers within an organization; users and systems that try to access the data must clearly be authorized to have such access. These roles are defined based on the confidentiality, integrity, and availability policies with which the digital information must comply. The purpose for the role based access policies are simple: if a user changes information that he/she is not authorized to change, then the integrity and confidentiality of the information is compromised. If a system cannot retrieve information that is needed, then the availability is compromised. If proprietary information is disclosed to persons who are not authorized to have access, then the confidentiality of the information is compromised. Corporations must establish role-based access controls for their information to retain all three qualities: confidentiality, integrity, and availability [4].

2.3.3.2 Payment Card Industry Data Security Standard

The credit card industry in 2006 released the Payment Card Industry Data Security Standard. This standard placed many restrictions on organizations and corporations that accept credit cards as a form of payment. If organization or corporations do not comply with said standard, their status as credit card processor could be revoked and the corporations could be fined. Some of the restrictions include the following: build and maintain a secure network, protect cardholder data, maintain a vulnerability management program, implement strong access control measures, regularly monitor and test networks, and maintain an information security policy [8].

2.4 Current Research in the Profession

Microsoft and Compuware worked together to perform a study of security practices in the United States and Europe. On October 9, 2006, they released the results of their study titled: *How Secure is Your Application Development?* Their claim is that: "security is only as good as the weakest link" [22]. They analyzed the completed breakdown of a web-application to show that the weakest link is the development of the application. Looking at the protection levels of a web-application there can be five areas that security needs to be in place: desktop layer, transport layer, access layer, network layer and application layer.

The desktop layer is where the end user is located. He or she decides to access the web-application. At the desktop layer the end user would be performing his or her part in the security process by having an anti-virus program fully operational and up-to-date. The next layer during the process of the end user accessing the web-application presents the transport layer. The transport layer is represented by the World Wide Web. The security measure at this stage in the process is an encrypted connection. During the access layer a firewall verifies that the communication passing through it to the web-application is an authentic connection. An intrusion detection system would be deployed to monitor the network layer. The user has now reached the application layer. This layer has been developed and placed on the web for viewing. Therefore the only security measures now in place are the built-in application protections.

The problem with relying on built in application protection is that most developers either don't understand the security requirements or they see security requirements as a limiting agent on the application [22]. Developers generally see security as a means by which to slow the application down, or not provide the access that the developer feels the

application deserves. Compuware and Microsoft both claim that the weakest link in this example is the application itself. "… security vulnerabilities at the application level are a form of design or coding defect…" [22]. Compuware and Microsoft have released a series of security approaches for developers. These include: assess business risk, develop the right architecture, code securely, test early and often, and validate security.

Compuware and Microsoft felt that if everyone was deploying anti-virus programs, firewalls, intrusion detection systems, then the weakest link had to be the application. They claim that with all these security measures in place, there should be no security breaches, but security breaches still occur based on commonly exploited attack mechanisms: SQL injections and buffer overflows [22]. These vulnerabilities in the software place the application in danger of being attacked once the information is made aware to the public. Compuware and Microsoft called for software developers to take security measures during the design phase to mitigate these risks [22].

2.5 Relevant Standards

There are numerous standards that are currently being referenced by organizations that use sensitive information. Sections 2.5.1-2.5.3 provides a description of three standards that can affect the software requirements specifications. As discussed later in this document, the software engineers developing the software requirements specifications must understand how the customer's organization needs to handle the data for their organization as well as how to handle the development of the SRS. For example, ISO/IEC 27001:2005 and NIST SP 800-100 both provide asset classification. Assets, such as data, are defined in the organizations information security management policies. These policies also provide specifics on how the data is to be handled. The software requirements specifications need to reflect a software design that will conform to the handling of such data according to the organizations information security management policies.

2.5.1 ISO/IEC 9001:2000

ISO/IEC 9001 provides requirements for quality management. It provides development companies an organized guidance to create a quality management system. The goal of a quality management system is to provide the developing organization a set of steps to developing a project and measurable guidelines to ensure that the customer receives a high quality product [26]. Software engineering companies would use ISO/IEC 9001 to provide a structure for developing software. A specific stage in this process might include develop software requirements specifications using IEEE Standard 830.

ISO/IEC 9001:2000 addresses issues such as: how to control documents, how to control records, how to perform internal audits, how to control nonconforming products, how to take corrective actions, and how to take preventative actions. All six of these categories, addressed by ISO/IEC 9001:2000, provide the quality management controls needed to maintain the software requirements specifications.

2.5.2 ISO/IEC 27001:2005

Information Security Management Policies are becoming a common practice. ISE/IEC 27001:2005 provides a template for developing an information security management system (ISMS). Located in the framework of an ISMS are: risk assessment and treatment, security policy, organization of information security, asset management, human resources security, physical and environmental security, communications and operations management, access control, information system acquisition, development, and maintenance, information security incident management, business continuity management, and compliance [27]. The software requirements specifications need to include references to the information security management policies of an organization. Located in the ISMS is detailed information about how assets are analyzed and protected inside the organization.

2.5.3 NIST SP 800-100

The NIST information security management standard contains the following aspects: information security management governance, system development life cycle, awareness and training, capital planning and investment control, interconnecting systems, performance measures, security planning, information technology contingency planning, risk management, certification, accreditation, and security awareness, security services and products acquisition, incident response, and configuration management. All of these policies, once created within an organization, provide detailed instructions on certain business aspects are to be addressed [25]. For example, in the awareness and training policy, a set of

specific guidelines will be specified as to how the training of a new system or security policy will be conducted inside the organization. In the risk management policy, an organization would specify specific risks that it feels could harm the organizations wellbeing. An example of such risk would be the risk of an unauthorized release of sensitive information. It would provide a classification of the risk and possible ways to mitigate the issue. Knowing what an organization believes are risks, during the development of the software requirements specifications for the organizations software, is a benefit software engineers will need to exploit. The design of the new system can ensure that these risks are either mitigated or eliminated.

3 SECURING SENSITIVE INFORMATION

Software engineers and computer scientists have progressively changed their focus when creating new software. When software was first being written it was focused on scientific and mathematical problems that could be solved more easily by a computer than by hand. Machine code was very tedious and difficult to write, with respect to today's programming languages. The focus during the beginning of computer programming was ensuring that the program completed the task accurately. Once accuracy was achieved, programmers began focusing on making their code more efficient due to insufficient hardware resources, due to cost. When the cost of hardware became low, programmers focused on developing large scale systems to make the lives of humans easier by automating tasks that would generally be tedious to users. Now that computers are so widely used throughout humans' lives, a new aspect of computer software has come into the light. This aspect is security.

3.1 The Need for Secure Data

On March 29, 2007, *The Boston Globe* reported that TJX had reported that 45.7 million credit and debit card numbers were stolen during a security breach [18]. This is the largest security breach publicly recorded. This security breach has already cost TJX over $5 million, and the cost is expected to continually rise. With a cost estimated at $90 per record stolen, the potential expense that TJX will have to spend estimates at nearly $4.1 billion dollars [19].

On April 7, 2007, *NetworkWorld* reported that the Chicago Public School system had issued a bulletin stating that two laptops had been stolen from their organization. Contained on the two laptops was nearly 40 thousand current and past employees' personally identifiable information. The information compromised in this case was names and social security numbers. A $10 thousand dollar reward has been offered for the arrest and conviction of the felon who stole the information [20]. At the same $90 per record stolen, the potential expense that the Chicago Public School system may have to spend to resolve the issue is approximately $3.6 million dollars [19].

Darwin Professional Underwriters performed a research study based on news reports and survey groups to provide corporations with a calculator to estimate the possible cost of a security breach. Darwin's calculator estimates approximately $166.20 per record breached. The costs calculated into the overall cost includes: internal investigation, notification/crisis management, and regulatory/compliance. Figure 5 provides a graphical representation based on Darwin's calculator [21].

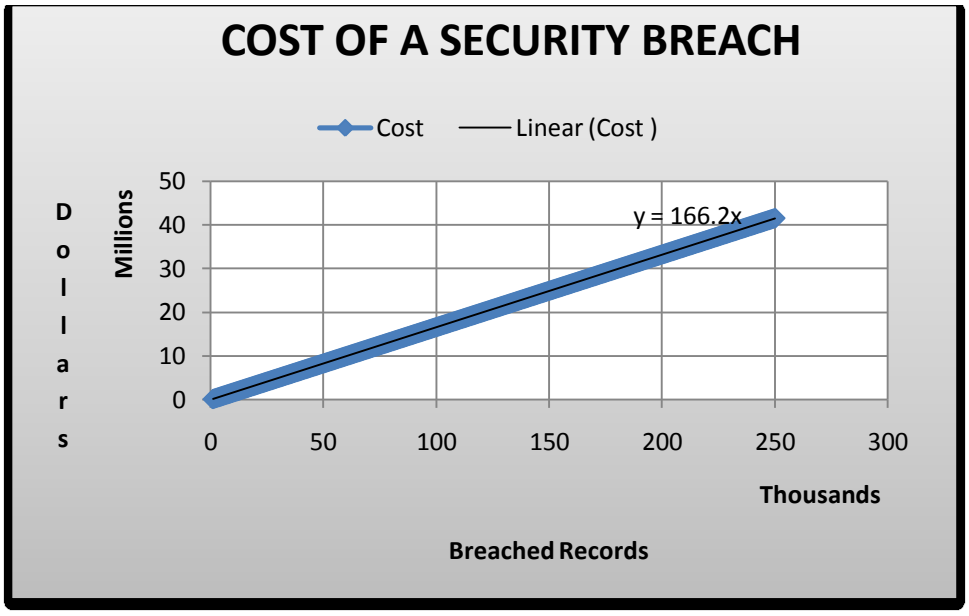

Figure 5: Cost of a Security Breach [21]

3.2 Digital Information and the Internet

Digital information being stored and transmitted throughout the world, via the Internet, includes items such as: medical information, credit card numbers, social security numbers, and recently, biometric information. As computer users become more accustomed to the digital world, more and more personal information will be stored in databanks of financial institutions, academic institutions, private organizations, governments, and corporations. During the creation of the Internet, security was not a high concern, for the only groups that had access to it were trusted government and educational entities. When the Internet became public domain and began to be used for commercial purposes, the need for security began to rise. The more persons that have access to a resource the less secure it becomes. Predators, thieves, and other criminals begin to find ways to exploit the new technology resources to advance their causes.

Computer software is not only a desktop application, which initial computer users were accustomed, but also a means by which to share information through large, multiregional corporations and entities, via the Internet.

Consumers trust their financial institutions to keep the personal information provided to them private, but if the institution is sending information across the Internet, is the information protected? During the software design process, if the requirements specifications for the software were to encrypt the data, then yes, but what if the specific security measures that needed to be put into place were not understood by developers?

3.3 IEEE Standard 830 Analysis

IEEE Standard 830 provides a template that is suggested to software engineers and computer scientists for use when developing software. The standard provides a location in the template to describe the security with which the system needs to comply. Even this standard has taken the afterthought approach to security. A generic description of the security requirements can easily be misinterpreted. Note that the standard does not insist that development organizations provide reasoning for the security of the system.

3.3.1 IEEE Standard 830 Security Analysis

United States' legislation is currently challenging the information technology profession to ensure that personal information is protected. By modifying the IEEE Standard 830 to include a section called security requirements, software engineers and computer scientists could obtain a better understanding of what security measures need taken in the software they are developing. What should the security requirements section contain? The security requirements section should begin by specifying factors that "protect the software from accidental and malicious access, use, modification, destruction, or disclosure" [13]. Notice that this is exactly what the IEEE Standard 830 insists is in the general security section that it provides. Following this description, it should provide a list of legislation, regulations, policies, or standards that could affect the corporation if the organization would experience an incident while using the software. Along with each piece of legislation, regulation, policy, or standard, a description of the statute or regulatory rule should be described. This section may need to be completed in conjunction with legal staff for either the developing company or the customer requesting the software. Another addition to the security section includes the organization's classification of their digital information and the specific requirements with which each classification must comply.

The legislation, regulations, policies, or standards should be provided by the organization requesting the software, for these organizations have a better understanding as to what regulations by which they have to abide. Software engineers should work with the requesting organization to ensure that all the details of these regulatory statutes are understood.

Another addition to the standard to help developers grasp a better understanding requires how the digital information should be handled is to be completed by placing a security section in each function description. This security section would list the following items: how the digital information the function is processing is classified inside the organization, how the information should be handled, and a reference to any regulatory standards that could affect the processing, storage, or transmission of such data – in the newly created security section of the software requirements specifications.

By providing this information to the developers of new software systems, developers have all the knowledge they need to complete a sound design, rather than adding patches to fix the problem after the software has been released.

3.3.2 Recommended Additions to IEEE Standard 830

Sections 3.3.2.1 and 3.3.2.2 show the recommended additions the standard with descriptions of what each section specifies.

3.3.2.1 SRS: Security Requirements Section

3.1. Security Requirements

It should be used to specify compliance regulations and policies as well as define the organizations data classification.

3.1.1. Data Classifications

This is a suggested addition to the standard. It would include information based on the requesting organizations data classifications based on their information security policies.

3.1.1.1. Classification Levels

This would define the levels of classifications and what actions must be performed to protect the data section. This will help the development team to accurately manage the digital information in the software.

3.1.2. Compliance Regulations

This is a suggested addition to the standard. This section provides and overall view of what regulations or policies the software must conform.

3.1.2.1. Regulation Name

This would be the actual name of the regulation.

3.1.2.1.1. Reference to Regulation

This section would provide information for researching the regulation.

3.1.2.1.2. Regulation Description

This section would provide a detailed description of the aspects of the regulation or policy that could affect the software design.

3.3.2.2 SRS: Security Section

3.1.1.8. Security

This section will be used to provide information to the developer about the handling of the data based on above suggested addition to the standard.

3.1.1.8.1. Regulatory Statutes

This section states the statute that could affect the design of the function.

3.1.1.8.2. Data Classification

This section states the classification of the data being handled by the function.

3.1.1.8.3. Data Handling

This section specifies the specific means to manipulate the data during processing to abide by the regulatory statute.

3.4 Sample Security Elicitation Questions

There are many different ways to elicit security requirements. One possible way, if using the unified modeling language, is by taking the use case diagrams developed during the specifying of the functional requirements of the system and changing them into misuse case diagrams. To do this, the diagrams are used to display what a user would not want to occur during the scenario being documented. Below is a set of example security elicitation questions that will help developers gain an understanding of the current security needs of their customer.

- Does your organization have to comply with any specific regulations or corporate policies?
- Would you provide us with a copy of these regulations or corporate policies?
- Do you currently have an information security management policy?
- If so, what data is classified inside your organization?
- How is this data classified?
- Are there any specific requirements for how the data shall be handled (for example: storage, transmission, processing, et cetera)?
- What security measures do you currently employ in your organization?
- Do you know or have a recommendation for the types of security that shall be used throughout the design of the new system?
- Do you currently own a VeriSign Certificate, or any other digital certificates?
- What business practices need to involve security?
- What aspects of the system being designed do you foresee needing security?
- How are you currently implementing user access controls?

3.5 Sample Requirements Elicitation

For instance: Developers are informed by their customer that they need to be able to process credit cards in their software. The customer also states that they need the ability to store the credit card information for future purchases of their customers. The developer would ask the following set of questions to correctly specify the functionality of the software: Are there any specific legislation, regulations or corporate policies pertaining to how credit card information is handled? The customer would then reply, yes, our organization has to comply with the Payment Card Industry Data Security Standard, also known as the PCIDSS. For the purpose of this example, it is assumed that this is the only regulatory statute with which the organization needs to comply. The developer would then ask, assuming that the developer already understands the organization's digital information classification and the requirements it must meet, how is the credit card information classified? The organization representative replies, the information is classified as red – the highest level of classification in the organization. The developer then asks, are there any specific ways that this information has to be handled? The organization representative then replies, it must be encrypted at all times possible and the complete number should never be displayed to any personnel within our organization.

3.5.1 Sample Security Requirements

The software requirements specifications security section would appear as follows:

<<ALL OTHER SPECIFICATIONS FROM SECTION 6.1>>

3.1 Security Requirements:

<<ALL TEMPLATE FIELDS FROM SECTION 6.1>>

0 Data Classifications

0 Red: Highest level of classification. This data should be encrypted using X standard. This classification holds information including: credit card information, <<ALL OTHER INFORMATION IN THIS CLASSIFICATION>>.

<<ALL OTHER DATA CLASSIFICATIONS>>

3.1.2 Compliance Regulations:

3.1.2.1 Payment Card Industry Data Security Standard, PCIDSS

3.1.2.1.1 Reference to Regulation: See reference 1.1.1 in the references section.

3.1.2.1.2 Description: PCIDSS is a regulatory statute placed on organizations and corporations that accept credit cards as a form of payment. It states that when displaying credit card numbers either on printed receipts or on the organizations user displays that only one of the following three items can be displayed: the first four numbers, the last four numbers, or both.

<<ALL OTHER COMPLIACE REGULATIONS>>

<<ALL OTHER SPECIFICATIONS FROM SECTION 6.1>>

3.5.2 Sample Function Specification

The software requirements specification for the previously mentioned example would appear as follows:

<<ALL OTHER SPECIFICATIONS FROM SECTION 6.1>>

3.3.1 FUNCTION X Specification

<<ALL TEMPLATE FIELDS FROM SECTION 6.1 >>

3.3.1.8 Security

3.3.1.8.1 Regulatory Statutes: PCIDSS further defined in section X.X

3.3.1.8.2 Data Classifications: Credit Card Number – Red

3.3.1.8.3 Data Handling: The credit card number should not be displayed to anyone in the organization. After Credit Card number is read into the system encrypt the information and store it into a masked field in the database. Ensure that the hard drive the information is stored on is encrypted using X standard.

<<ALL OTHER SPECIFICATIONS FROM SECTION 6.1>>

3.6 Technique for Specification of Data

ISO/IEC 11179-1 is a standard that is used to specify information about data (metadata) [23]. This specification of the metadata is to be stored in a metadata registry (MDR). The purpose of the standard is to specify data so that it can be shared in a standard way across distributed or large scale systems. By using ISO/IEC 11179-1, software engineers can ensure that the data is being represented by a specific set of rules [23].

The data elements are classified by placing them in a conceptual domain. A conceptual domain is further divided into a set of categories – representation of the meaning and permissible values [23]. By using the customers data classification based on their organizations information security management policies, software developers can specify the necessary information needed to utilize an MDR. This information serves as a framework for what they data looks like and should be handled. An example based on the previously mentioned scenario follows.

3.6.1 Sample Classification of Data

Conceptual Domain Name:	CreditCards
Conceptual Domain Definition:	Has a set of digits between 13 and 16
Conceptual Security Policy:	Only the last 4 digits can be displayed in the system.

Value Domain Name (1):	MasterCard
Value Domain Description:	Card prefix must be between 51-55 and have a total of 16 digits
Value Domain Name (2):	Visa
Value Domain Description:	Card prefix must be 4 and have either 13 or 16 digits

3.7 IEEE Standard 830 and the Law

Most of the state security breach laws list specifically that organizations and corporations must inform customers "…whose unencrypted personal information was, or is reasonably believed to have been, acquired by an unauthorized person" [17]. If developers are aware of this clause in the state legislations, they could develop software that would automatically encrypt information prior to storage or transmission and decrypt it upon processing. This would minimize the risk of disclosing personal information. Taking extra measurers to ensure that the software is more secure will make the cost of the product more expensive – more requirements, more elicitation, more coding, and more bandwidth – but it will save the company from a long and involved legal battle, due to disclosure of information under a legislative regulation that requires the information secured.

Section 6.1 shows the aforementioned recommended changes to the software requirements specifications outline.

4 CONCLUSION

Securing information in software engineering projects is becoming increasingly necessary. Many United States federal and state governments are enacting legislation to ensure that digital information provided to financial institutions is protected. Corporations also have to set their own policies and standards to ensure information that they need to complete business is secure. An excellent example is the previously mentioned PCIDSS.

The corporations and governments that are regulating how digital information is handled are relying on the information technology professionals, including computer scientists and software engineers, to ensure that their regulations are upheld and audited. As new software projects are defined and software requirements specifications are gathered, more emphasis needs placed on security throughout the design phase, rather than just at the end or from a very low level of security.

This paper presented a proposed change to the software requirements specifications outline provided by IEEE Standard 830. This change would help ensure that security is analyzed in an earlier stage of the software development lifecycle. The new template will help the information technology industry to develop more secure and legally compliant software.

5 FUTURE WORK

This document will be provided to the Secretary, IEEE-SA Standards Board, as a suggested change. After this document is presented to the board, the board may make a decision to either: create a new standard, create a revision to the current standard, amend the current standard, correct any technical issues of the current standard, correct grammatical errors in the current standard, or do nothing.

IEEE has set a specific set of guidelines that must be followed to invoke a change to a standard. First, a project authorization request must be filed to the New Standards Committee (NesCom). Once approved by NesCom a working group will be developed. The working group is charged with the task of developing a draft. After the draft is complete, the sponsor of the working group will ballot the draft standard. If the ballot is successful, then the draft is sent to the IEEE Review Committee (RevCom). RevCom will make a recommendation to the IEEE-SA Standards Board. After the Standards Board has approved the new standard, it enters the manage phase. The first step of the manage phase is to publish the standard. Once published, it will be reviewed every five years for relevance [24].

6 APPENDICES

6.1 APPENDIX A: Software Requirements Specifications

The following is a suggested requirements specifications template. The modified sections of the IEEE Standard 830-1998 are **highlighted** [13]. There are many ways to organized section 3 of the template provided in IEEE Standard 830-1998 and they are located in section 6.2.

1. Introduction

This section provides an overview of the entire SRS

1.1. Purpose

This section specifies the intended audience and provides the purpose of the SRS

1.2. Scope

Identifies the software products being developed by name and provides a brief description as to what each of the products will or will not do. This section also provides the benefits and objectives of the developing software.

1.3. Definitions, Acronyms, and Abbreviations

This section provides information that is needed to correctly interpret the SRS.

1.4. References

This section provides a list of all sources used to create the document or the citations for any documents that are referenced throughout the SRS.

1.5. Overview

Describes what the rest of the SRS contains. ***Ensure that in this section a description of how the security information is presented in the SRS is described.***

2. Overall Description

This section describes factors that affect the product or the SRS.

2.1. Product Perspective

The product perspective relates the developing product to other products. It also specifies how the system operates inside various constraints.

2.1.1. System Interfaces

This section lists the system interfaces and the functionality of the software to accomplish the system requirement.

2.1.2. User Interfaces

This describes both the logical characteristics of each interface to the user and the aspects of optimizing the interface with the person who will be using the system.

2.1.3. Hardware Interfaces

This will provide protocols and supported devices for the developing system. It also provides the configuration characteristics between the software and hardware.

2.1.4. Software Interfaces

This provides information on how the developing software will connect to other software products necessary. Items needed to specify a software connection are: name, mnemonic, specification number, version number, and source. A brief discussion should be provided as to the reasoning for the connection to the other software product.

2.1.5. Communications Interfaces

This provides information on the various communication protocols the developing software will interface.

2.1.6. Memory

This specifies the limits on primary and secondary memory.

2.1.7. Operations

This specifies the normal and special operations required by the user.

2.1.8. Site Adaptation Requirements

This provides information on the environment and mission of the site where the software is being installed. It would provide special requirements necessary for the specific location.

2.2. Product Functions

This provides a summary of the major functionality within the system. Textual and graphical methods to specifying the functionality of the software is encouraged.

2.3. User Characteristics

This provides a general description of the system users: technical expertise, education level, language, or experience.

2.4. Constraints

*This includes information that would limit the developer's options. The following subheadings (~~**Regulatory Policies**~~2.4.1-2.4.11) are some possible constraints that may need considered.*

~~**2.4.1. Regulatory Policies**~~

~~***This describes corporate regulations that would limit the developer's options.***~~

2.4.2. Hardware Limitations

This provides descriptions of any hardware limitations.

2.4.3. Interfaces to Other Applications

This describes interfaces to commercial off the shelf systems as well as other previously developed systems.

2.4.4. Parallel Operations

This describes any required parallel operations the system may need to perform.

2.4.5. Audit Functions

This describes any required audit or monitoring function necessary.

2.4.6. Control Functions

This describes any specific control functions that could limit the developer's options.

2.4.7. Higher-order Language Requirements

This describes specific language constraints due to the language of the system.

2.4.8. Signal Handshake Protocols

For example: ACK-NACK or XON-XOFF.

2.4.9. Reliability Requirements

This describes any specific reliability requirements.

2.4.10. Criticality of the Applications

This describes the criticality of the system being developed.

2.4.11. Safety ~~**and Security**~~ Considerations

This provides an overview of any known safety ~~***or security***~~ *issues that would need to be known during the development phase.*

2.5. Assumptions and Dependencies

This provides a list of factors that affect the requirements stated in the SRS. These are not design constraints but any changes to these items would inflict a necessary change to the SRS.

2.6. Apportioning of Requirements

This section identifies requirements that might be delayed for future versions or releases.

3. Specific Requirements

This section is to define the specific technical details of the system so that designers can develop the product and testers can test the product.

3.1. Security Requirements

It should be used to specify compliance regulations and policies as well as define the organizations data classification. This section provides overall security requirements for the system. Sections 3.1.3-0 are some recommended evaluated areas, there are many others that could be listed in this section.

3.1.1. Data Classifications

This is a suggested addition to the standard. It would include information based on the requesting organizations data classifications based on their information security policies.

3.1.1.1. Classification Levels

This would define the levels of classifications and what actions must be performed to protect the data section. This will help the development team to accurately manage the digital information in the software.

3.1.2. Compliance Regulations

This is a suggested addition to the standard. This section provides and overall view of what regulations or policies the software must conform.

3.1.2.1. Regulation Name

This would be the actual name of the regulation.

3.1.2.1.1. Reference to Regulation

This section would provide information for researching the regulation.

3.1.2.1.2. Regulation Description

This section would provide a detailed description of the aspects of the regulation or policy that could affect the software design.

3.1.3. Utilize Certain Cryptographical Techniques

This section would provide the specific technique or encryption standard to be utilized during development.

3.1.4. Keep Specific Log or History Data Sets

This section would specify what information needs log and the length of the logs (space or time).

3.1.5. Assign Certain Functions to Different Modules

This section would separate the functions into groups based on security level or access level.

3.1.6. Restrict Communications between some Areas of the Program

This section would state where communication paths should be restricted.

3.1.7. Check Data Integrity for Critical Values

This would specify algorithm to be used to compute checksums and what aspects need checksums.

3.2. External Interfaces

This section provides a detailed description of all inputs into and outputs from the system. Section 3.2.1 provides a breakdown as to specifying the data inputs/outputs.

3.2.1. Name of Item

This contains the name of the input/output.

3.2.1.1. Description of Purpose

This section would describe why the input/output is needed.

3.2.1.2. Source of Input or Destination of Output

This section would state where the input is coming or where the output is going.

3.2.1.3. Valid Range, Accuracy, and/or Tolerance

This section would set threshold values of the input/output.

3.2.1.4. Units of Measure

This would specify what units the input/output is in.

3.2.1.5. Timing

This would set threshold value for the length of time to receive the input or provide the output.

3.2.1.6. Relationships to other inputs/outputs

This would describe how it interacts with other inputs/outputs.

3.2.1.7. Screen Formats/Organization

This section is to describe how the screen should be organized.

3.2.1.8. Window Formats/Organization

This section is to describe how the window should be organized.

3.2.1.9. Data Formats

This section defines the format or type of the input.

3.2.1.10. Command Formats

This section defines how the information is received/provided.

3.2.1.11. End Messages

This section defines the final state or message after processing the data.

3.3. Functions

This section is used to specify the functions in the software product.

3.3.1. Function Name

This section specifically states the function name as it would appear in the code.

3.3.1.1. Validity Checks on the Inputs

This section states what checks shall be performed on all inputs into the function.

3.3.1.2. Exact Sequence of Operations

This section defines the steps of the function.

3.3.1.3. Responses to Abnormal Situations

This section defines how the system should handle abnormal conditions. Sections 3.3.1.3.1-3.3.1.3.3 are some recommended conditions to evaluate; there are many others that could be added to this section.

3.3.1.3.1. Overflow

This section states how the system should handle an overflow issue.

3.3.1.3.2. Communication Facilities

This section states how the system should handle communication faults.

3.3.1.3.3. Error Handling and Recovery

This section describes specific error conditions and how the system should recover. These will be specific to each system.

3.3.1.4. Effect of Parameters

This section should specify what each parameter's purpose is in the function.

3.3.1.5. Relationship of Outputs to Inputs

This section is used to show how the information is converted from an input to an output.

3.3.1.6. Input/output Sequences

Provides the sequences by which to receive or produce an input/output.

3.3.1.7. Formulas for Input to Output conversion

This section provides specific formulas for converting the input to an output.

3.3.1.8. Security

This section will be used to provide information to the developer about the handling of the data based on above suggested addition to the standard.

3.3.1.8.1. Regulatory Statutes

This section states the statute that could affect the design of the function.

3.3.1.8.2. Data Classification

This section states the classification of the data being handled by the function.

3.3.1.8.3. Data Handling

This section specifies the specific means to manipulate the data during processing to abide by the regulatory statute.

3.4. Performance Requirements

This section is used to provide performance requirements to the developer during the coding phase. It provides numerical requirements placed on the software or on human interaction with the software as a whole.

3.4.1. Static Numerical Requirements

These are values that are set that should not change.

3.4.1.1. Number of Terminals to be Supported

This section provides the number of terminals that the software will operate on.

3.4.1.2. Number of Simultaneous Users to be Supported

This section provides the number of users the system should be able to support.

3.4.1.3. Amount and Type of Information to be Handled

This section provides information on the amount of information and the type of information that the system will be processing.

3.4.2. Dynamic Numerical Requirements

These are values that are based on threshold values or a function of time.

3.4.2.1. Number of Transactions to be Processed in a Given Time Period

This provides the number of transaction to be processed and the time they have to be processed in.

3.5. Logical Database Requirements

This section provides the requirements of anything to be placed or access a database. Sections 3.5.1-3.5.6 are some suggested areas to consider when specifying database requirements.

3.5.1. Types of Information used by Various Functions

This section specifies the types of data being used.

3.5.2. Frequency of Use

This specifies how frequently the database will be used.

3.5.3. Accessing Capabilities

This specifies how the functions will access the database.

3.5.4. Data Entities and their Relationships

This specifies what entities are located in the database and how they are related to each other.

3.5.5. Integrity Constraints

This sets the requirements on how the database verifies that the information is correct.

3.5.6. Data Retention Requirements

This specifies how long the data is to be kept.

3.6. Standards Compliance

This section specifies the developer's standards for developing the software. This is specified to ensure consistency.

3.6.1. Report Format

This specifies how the developers will provide reports to the customers and what is to be located in them.

3.6.2. Data Naming

This section specifies the standard by which information is named in the source code.

3.6.3. Accounting Procedures

This section specifies how functions will call each other.

3.6.4. Audit Tracing

This specifies how to trace processes that have occurred in the system.

3.7. Software System Attributes

These are requirements that have not been elsewhere documented that the system must conform. Sections 3.7.1-3.7.5 provides a list of suggested areas to evaluate. There are many other evaluation methods that could be listed in this section.

3.7.1. Reliability

This section specifies how reliable the software must be at the time of delivery.

3.7.2. Availability

Specifies when the system should be available. It can analyze checkpoints, recoveries, and restarts.

~~3.7.3. Security~~

~~This section provides overall security requirements for the system. Sections 0-3.7.3.5 are some recommended evaluated areas, there are many others that could be listed in this section.~~

~~3.7.3.1. Utilize Certain Cryptographical Techniques~~

~~This section would provide the specific technique or encryption standard to be utilized during development.~~

~~3.7.3.2. Keep Specific Log or History Data Sets~~

~~This section would specify what information needs log and the length of the logs (spaee or time).~~

~~3.7.3.3. Assign Certain Functions to Different Modules~~

~~This section would separate the functions into groups based on security level or access level.~~

~~3.7.3.4. Restrict Communications between some Areas of the Program~~

~~This section would state where communication paths should be restricted.~~

~~3.7.3.5.~~ **~~Check Data Integrity for Critical Values~~**

~~This would specify algorithm to be used to compute checksums and what aspects need checksums.~~

3.7.4. Maintainability

This specifies requirements that relate to the ease of maintenance. There may be some requirement for certain modularity, interfaces, complexity, et cetera.

3.7.5. Portability

This section defines how portable the system must or should be.

3.7.5.1. Percentage of Components with Host-dependent Code

This is a threshold percentage based on total components.

3.7.5.2. Percentage of code that is host dependent

This is a threshold value based on all of the system code.

3.7.5.3. Use of a Proven Portable Language

This section specifies the use of a particular language that the code is to be written in.

3.7.5.4. Use of a Particular Compiler or Language Subset

This section specifies the use of a particular compiler for the code.

3.7.5.5. Use of a Particular Operating System

This section specifies what operating systems the software should be able to operate on.

6.2 APPENDIX B: SRS Section 3 Templates

All of the following templates have been modified based on the templates located in IEEE Standard 830-1998 [13].

6.2.1 Organized by Mode [13]

3. Specific Requirements
    - 3.1. External Interface Requirements
        - 3.1.1. User Interfaces
        - 3.2.1. Hardware Interfaces
        - 3.3.1. Software Interfaces
        - 3.4.1. Communications Interfaces
    - **3.2. Security Requirements**
        - **3.2.1. Data Classifications**
            - **3.2.1.1. Classification Levels**
        - **3.2.2. Compliance Regulations**
            - **3.2.2.1. Regulation Name**
                - **3.2.2.1.1. Reference to Regulation**
                - **3.2.2.1.2. Regulation Description**
        - **3.2.3. Other Security Requirements**
    - 3.3. Functional Requirements
        - 3.3.1. Mode 1
            - 3.3.1.1. Functional Requirement 1.1
                - .
                - .
                - .
                - **3.3.1.1.x. Security**
                    - **3.3.1.1.x.1. Regulatory Statutes**
                    - **3.3.1.1.x.2. Data Classification**
                    - **3.3.1.1.x.3. Data Handling**
            - .
            - .
            - .
            - 3.3.1.n. Functional Requirement 1.n
                - .
                - .
                - .
                - **3.3.1.n.x. Security**
                    - **3.3.1.n.x.1. Regulatory Statutes**
                    - **3.3.1.n.x.2. Data Classification**
                    - **3.3.1.n.x.3. Data Handling**
        - 3.3.2. Mode 2
        - .
        - .
        - .

3.3.m. Mode m

3.3.m.1. Functional Requirement m.1

.

.

.

3.3.m.1.x. Security

3.3.m.1.x.1. Regulatory Statutes

3.3.m.1.x.2. Data Classification

3.3.m.1.x.3. Data Handling

.

.

.

3.3.m.n. Functional Requirement m.n

.

.

.

3.3.m.n.x. Security

3.3.m.n.x.1. Regulatory Statutes

3.3.m.n.x.2. Data Classification

3.3.m.n.x.3. Data Handling

3.4. Performance Requirements

3.5. Design Constraints

3.6. Software System Attributes

3.7. Other Requirements

6.2.2 Organized by Mode: Version 2 [13]

3. Specific Requirements
    3.1. Functional Requirements
        3.1.1. Mode 1
            3.1.1.1. External Interfaces
                3.1.1.1.1. User Interfaces
                3.1.1.1.2. Hardware Interfaces
                3.1.1.1.3. Software Interfaces
                3.1.1.1.4. Communications Interfaces
            3.1.1.2. Security Requirements
                3.1.1.2.1. Data Classifications
                    3.1.1.2.1.1. Classification Levels
                3.1.1.2.2. Compliance Regulations
                    3.1.1.2.2.2. Regulation Name
                        3.1.1.2.2.2.1. Reference to Regulation
                        3.1.1.2.2.2.2. Regulation Description
                3.1.1.2.3. Other Security Requirements
            3.1.1.3 Functional Requirements
                3.1.1.3.1. Functional Requirement 1
                    .
                    .
                    .
                    3.1.1.3.1.x. Security
                        3.1.1.3.1.x.1. Regulatory Statutes
                        3.1.1.3.1.x.2. Data Classification
                        3.1.1.3.1.x.3. Data Handling
                .
                .
                .
                3.1.1.3.n. Functional Requirement n
                    .
                    .
                    .
                    3.1.1.3.1.x. Security
                        3.1.1.3.1.x.1. Regulatory Statutes
                        3.1.1.3.1.x.2. Data Classification
                        3.1.1.3.1.x.3. Data Handling

3.1.1.4 Performance

3.1.2. Mode 2

.

.

.

3.1.m. Mode m

3.2 Design Constraints

3.3 Software System Attributes

3.4 Other Requirements

6.2.3 Organized by User Class [13]

3. Specific Requirements
    3.1. External Interface Requirements
        3.1.1. User Interfaces
        3.1.2. Hardware Interfaces
        3.1.3. Software Interfaces
        3.1.4. Communications Interfaces
    3.2. Security Requirements
        3.2.1. Data Classifications
            3.2.1.1. Classification Levels
        3.2.2. Compliance Regulations
            3.2.2.1. Regulation Name
                3.2.2.1.1. Reference to Regulation
                3.2.2.1.2. Regulation Description
        3.2.3. Other Security Requirements
    3.3. Functional Requirements
        3.3.1. User Class 1
            3.3.1.1. Functional Requirement 1.1
                .
                .
                .
                3.3.1.1.x. Security
                    3.3.1.1.x.1. Regulatory Statutes
                    3.3.1.1.x.2. Data Classification
                    3.3.1.1.x.3. Data Handling
            .
            .
            .
            3.3.1.n Functional Requirement 1.n
                .
                .
                .
                3.3.1.n.x. Security
                    3.3.1.n.x.1. Regulatory Statutes
                    3.3.1.n.x.2. Data Classification
                    3.3.1.n.x.3. Data Handling
        3.3.2. User Class 2
        .
        .
        .

3.3.m. User Class m

3.3.m.1. Functional Requirement m.1

.

.

.

3.3.m.1.x. Security

3.3.m.1.x.1. Regulatory Statutes

3.3.m.1.x.2. Data Classification

3.3.m.1.x.3. Data Handling

.

.

.

3.3.m.n. Functional Requirement m.n

.

.

.

3.3.m.n.x. Security

3.3.m.n.x.1. Regulatory Statutes

3.3.m.n.x.2. Data Classification

3.3.m.n.x.3. Data Handling

3.4. Performance Requirements

3.5. Design Constraints

3.6. Software System Attributes

3.7. Other Requirements

6.2.4 Organized by Object [13]

3. Specific Requirements
    - 3.1. External Interface Requirements
        - 3.1.1. User Interfaces
        - 3.1.2. Hardware Interfaces
        - 3.1.3. Software Interfaces
        - 3.1.4. Communications Interfaces
    - **3.2. Security Requirements**
        - **3.2.1. Data Classifications**
            - **3.2.1.1. Classification Levels**
        - **3.2.2. Compliance Regulations**
            - **3.2.2.1. Regulation Name**
                - **3.2.2.1.1. Reference to Regulation**
                - **3.2.2.1.2. Regulation Description**
        - **3.2.3. Other Security Requirements**
    - 3.3. Classes/Objects
        - 3.3.1. Class/Object 1
            - 3.3.1.1. Attributes (direct or inherited)
                - 3.3.1.1.1. Attribute 1
                - .
                - .
                - .
                - **3.3.1.1.1.x. Data Classification**
            - .
            - .
            - .
            - 3.3.1.1.n. Attribute n
                - .
                - .
                - .
                - **3.3.1.1.n.x. Data Classification**

3.3.1.2. Functions (services, methods, direct or inherieted)

3.3.1.2.1. Functional Requirement 1.1

.

.

.

3.3.1.2.1.x. Data Handling

.

.

.

3.3.1.2.m. Functional Requirement 1.m

.

.

.

3.3.1.2.m.x. Data Handling

3.3.1.3. Messages (communications received or sent)

.

.

.

3.3.1.3.x. Regulatory Statutes

3.3.2. Class/Object p

.

.

.

3.3.p. Class/Object p

3.4. Performance Requirements

3.5. Design Constraints

3.6. Software System Attributes

3.7. Other Requirements

6.2.5 Organized by Feature [13]

3. Specific Requirements
    - 3.1. External Interface Requirements
        - 3.1.1. User Interfaces
        - 3.1.2. Hardware Interfaces
        - 3.1.3. Software Interfaces
        - 3.1.4. Communications Interfaces
    - **3.2. Security Requirements**
        - **3.2.1. Data Classifications**
            - **3.2.1.1. Classification Levels**
        - **3.2.2. Compliance Regulations**
            - **3.2.2.1. Regulation Name**
                - **3.2.2.1.1. Reference to Regulation**
                - **3.2.2.1.2. Regulation Description**
        - **3.2.3. Other Security Requirements**
    - 3.3. System Features
        - 3.3.1. System Feature 1
            - 3.3.1.1. Introduction/Purpose of feature
            - 3.3.1.2. Stimulus/Response sequence
            - 3.3.1.3. Associated Functional Requirements
                - 3.3.1.3.1. Functional Requirement 1
                - .
                - .
                - .
                - **3.3.1.3.1.x. Security**
                    - **3.3.1.3.1.x.1. Regulatory Statutes**
                    - **3.3.1.3.1.x.2. Data Classification**
                    - **3.3.1.3.1.x.3. Data Handling**
                - .
                - .
                - .
                - 3.3.1.3.n. Functional Requirement n
                - .
                - .
                - .
                - **3.3.1.3.n.x. Security**
                    - **3.3.1.3.n.x.1. Regulatory Statutes**
                    - **3.3.1.3.n.x.2. Data Classification**
                    - **3.3.1.3.n.x.3. Data Handling**

3.3.2. System Feature 2

.

.

.

3.3.m. System Feature m

.

.

.

3.4. Performance Requirements

3.5. Design Constraints

3.6. Software System Attributes

3.7. Other Requirements

6.2.6 Organized by Stimulus [13]

3. Specific Requirements
    - 3.1 External Interfaces
        - 3.1.1. User Interfaces
        - 3.1.2. Hardware Interfaces
        - 3.1.3. Software Interfaces
        - 3.1.4. Communications Interfaces
    - **3.2. Security Requirements**
        - **3.2.1. Data Classifications**
            - **3.2.1.1. Classification Levels**
        - **3.2.2. Compliance Regulations**
            - **3.2.2.1. Regulation Name**
                - **3.2.2.1.1. Reference to Regulation**
                - **3.2.2.1.2. Regulation Description**
        - **3.2.3. Other Security Requirements**
    - 3.3. Functional Requirements
        - 3.3.1. Stimulus 1
            - 3.3.1.1. Functional Requirement 1.1
                - .
                - .
                - .
                - **3.3.1.1.x. Security**
                    - **3.3.1.1.x.1. Regulatory Statutes**
                    - **3.3.1.1.x.2. Data Classification**
                    - **3.3.1.1.x.3. Data Handling**
            - .
            - .
            - .
            - 3.3.1.n. Functional Requirement 1.n
                - .
                - .
                - .
                - **3.3.1.n.x. Security**
                    - **3.3.1.n.x.1. Regulatory Statutes**
                    - **3.3.1.n.x.2. Data Classification**
                    - **3.3.1.n.x.3. Data Handling**
        - 3.3.2. Stimulus 2
        - .
        - .
        - .

3.3.m. Stimulus m

3.3.m.1. Functional Requirement m.1

.

.

.

3.3.m.1.x. Security

3.3.m.1.x.1. Regulatory Statutes

3.3.m.1.x.2. Data Classification

3.3.m.1.x.3. Data Handling

.

.

.

3.3.m.n. Functional Requirement m.n

.

.

.

3.3.m.n.x. Security

3.3.m.n.x.1. Regulatory Statutes

3.3.m.n.x.2. Data Classification

3.3.m.n.x.3. Data Handling

3.4. Performance Requirements

3.5. Design Constraints

3.6. Software System Attributes

3.7. Other Requirements

6.2.7 Organized by Functional Hierarchy [13]

3. Specific Requirements
    - 3.1. External Interface Requirements
        - 3.1.1. User Interfaces
        - 3.1.2. Hardware Interfaces
        - 3.1.3. Software Interfaces
        - 3.1.4. Communications Interfaces
    - **3.2. Security Requirements**
        - **3.2.1. Data Classifications**
            - **3.2.1.1. Classification Levels**
        - **3.2.2. Compliance Regulations**
            - **3.2.2.1. Regulation Name**
                - **3.2.2.1.1. Reference to Regulation**
                - **3.2.2.1.2. Regulation Description**
        - **3.2.3. Other Security Requirements**
    - 3.3. Functional Requirements
        - 3.3.1. Information Flows
            - 3.3.1.1. Data Flow Diagram 1
                - 3.3.1.1.1. Data Entities
                - 3.3.1.1.2. Pertinent Processes
                - 3.3.1.1.3. Topology
            - 3.3.1.2. Data Flow Diagram 2
            - .
            - .
            - .
            - 3.3.1.n. Data Flow Diagram n
                - 3.2.1.n.1. Data Entities
                - 3.2.1.n.2. Pertinent Processes
                - 3.2.1.n.3. Topology
        - 3.3.2. Process Descriptions
            - 3.3.2.1. Process 1
                - 3.3.2.1.1. Input Data Entities
                - 3.3.2.1.2. Algorithm or Formula of Process
                    - **3.3.2.1.2.1. Regulatory Statutes**
                    - **3.3.2.1.2.2. Data Handling**
                - 3.3.2.1.3. Affected Data Entities
            - 3.3.2.2. Process 2
                - 3.3.2.2.1. Input Data Entities
                - 3.3.2.2.2. Algorithm or Formula of Process
                    - **3.3.2.2.2.1. Regulatory Statutes**
                    - **3.3.2.2.2.2. Data Handling**
                - 3.3.2.2.3. Affected Data Entities
            - .
            - .
            - .

3.3.2.m. Process m
3.3.2.m.1. Input Data Entities
3.3.2.m.2. Algorithm or Formula of Process
3.3.2.m.2.1. Regulatory Statutes
3.3.2.m.2.2. Data Handling
3.3.2.m.3. Affected Data Entities
3.3.3. Data Construct Specifications
3.3.3.1. Construct 1
3.3.3.1.1. Record Type
3.3.3.1.2. Constituent Fields
3.3.3.2. Construct 2
3.3.3.2.1. Record Type
3.3.3.2.2. Constituent Fields
.
.
.
3.3.3.p. Construct p
3.3.3.p.1. Record Type
3.3.3.p.2. Constituent Fields
3.3.4. Data Dictionary
3.3.4.1. Data Element 1
3.3.4.1.1. Name
3.3.4.1.2. Representation
3.3.4.1.3. Units/Format
3.3.4.1.4. Precision/Accuracy
3.3.4.1.5. Range
3.3.4.1.6. Data Classification
3.3.4.2. Data Element 2
3.3.4.2.1. Name
3.3.4.2.2. Representation
3.3.4.2.3. Units/Format
3.3.4.2.4. Precision/Accuracy
3.3.4.2.5. Range
3.3.4.2.6. Data Classification
.
.
.
3.3.4.q. Data Element q
3.3.4.q.1. Name
3.3.4.q.2. Representation
3.3.4.q.3. Units/Format
3.3.4.q.4. Precision/Accuracy
3.3.4.q.5. Range
3.3.4.q.6. Data Classification

3.4. Performance Requirements
3.5. Design Constraints
3.6. Software System Attributes
3.7. Other Requirements

6.2.8 Showing Multiple Organizations [13]

3. Specific Requirements
    3.1. External Interfaces
        3.1.1. User Interfaces
        3.1.2. Hardware Interfaces
        3.1.3. Software Interfaces
        3.1.4. Communications Interfaces
    3.2. Security Requirements
        3.2.1. Data Classifications
            3.2.1.1. Classification Levels
        3.2.2. Compliance Regulations
            3.2.2.1. Regulation Name
                3.2.2.1.1. Reference to Regulation
                3.2.2.1.2. Regulation Description
        3.2.3. Other Security Requirements
    3.3. Functional Requirements
        3.3.1. User Class 1
            3.3.1.1. Feature 1.1
                3.3.1.1.1. Introduction/Purpose of Feature
                3.3.1.1.2. Stimulus/Response Sequence
                3.3.1.1.3. Associated Functional Requirements
                .
                .
                .
                3.3.1.1.3.x. Security
                3.3.1.1.3.x.1. Regulatory Statutes
                3.3.1.1.3.x.2. Data Classification
                3.3.1.1.3.x.3. Data Handling
            3.3.1.2. Feature 1.2
                3.3.1.2.1. Introduction/Purpose of Feature
                3.3.1.2.2. Stimulus/Response Sequence
                3.3.1.2.3. Associated Functional Requirements
                .
                .
                .
                3.3.1.2.3.x. Security
                3.3.1.2.3.x.1. Regulatory Statutes
                3.3.1.2.3.x.2. Data Classification
                3.3.1.2.3.x.3. Data Handling
            .
            .
            .

3.3.1.m. Feature 1.m

3.3.1.m.1. Introduction/Purpose of Feature

3.3.1.m.2. Stimulus/Response Sequence

3.3.1.m.3. Associated Functional Requirements

.

.

.

3.3.1.m.3.x. Security

3.3.1.m.3.x.1. Regulatory Statutes

3.3.1.m.3.x.2. Data Classification

3.3.1.m.3.x.3. Data Handling

3.3.2. User Class 2

.

.

.

3.3.n. User Class n

.

.

.

3.4. Performance Requirements

3.5. Design Constraints

3.6. Software System Attributes

3.7. Other Requirements

7 BIBLIOGRAPHY

[1] 104th Congress. 1996. Public Law 104-191. *Health Insurance Portability and Accountability Act of 1996.* [Online] August 21, 1996. [Cited: March 18, 2007.] http://aspe.hhs.gov/admnsimp/pl104191.htm.

[2] American National Standard Institute. 1984. *IEEE Guide to Software Requirements Specifications.* New York, NY : The Institute of Electrical and Electronics Engineers, Inc., 1984. ANSI/IEEE Std. 830-1984.

[3] Apani Networks. 2006. The California Security Breach Information Act (SB1386) and Its Impact on IT Security. *Apani Knowledge Center.* [Online] July 19, 2006. [Cited: March 17, 2007.] http://www.apani.com/pdf/Apani-sb1386.pdf.

[4] Barkley, John. 1995. Aspects of Security Policies. *Role Based Access Control.* [Online] National Institute of Standards and Technology, January 9, 1995. [Cited: March 18, 2007.] http://hissa.ncsl.nist.gov/rbac/paper/node2.html.

[5] Family Policy Compliance Office. 2005. Family Educational Rights and Privacy Act (FERPA). *U. S. Department of Education: Promoting educational excellence for all Americans.* [Online] February 17, 2005. [Cited: MArch 18, 2007.] http://www.ed.gov/policy/gen/guid/fpco/ferpa/index.html.

[6] Federal Trade Commission. The Gramm-Leach-Bliley Act: The Financial Privacy Rule. *Privacy Initiatives: Financial Privacy.* [Online] Federal Trade Commission. [Cited: March 17, 2007.] http://www.ftc.gov/privacy/privacyinitiatives/financial_rule.html.

[7] Institute for Electrical and Electronic Engineers. 2007. IEEE Standards. *IEEE The world's leading professional association for the advancement of technology.* [Online] IEEE, 2007. [Cited: March 17, 2007.] http://www.ieee.org/web/standards/home/index.html.

[8] MasterCard International Inc. and Visa U.S.A. Inc. 2006. Payment Card Industry (PCI) Data Security Standard. *Welcome to PCI Security Standards Council.* [Online] September 2006. [Cited: March 19, 2007.] https://www.pcisecuritystandards.org/pdfs/pci_dss_v1-1.pdf.

[9] Molsen. 2006. Chart of State Privacy and Data Security Rules. *Kauffman eVenturing.* [Online] August 17, 2006. [Cited: March 17, 2007.] http://eventuring.org/eShip/appmanager/eVenturing/ShowDoc/eShipWebCacheRepository/Documents/State_Privacy_and_Rules.xls.

[10] *Software Engineering Techniques.* NATO Science Committee. 1969. [ed.] J. N. Buxton and B. Randell. Rome, Italy : NATO Science Committee, 1969.

[11] Sommerville, Ian. 2007. *Software Engineering.* 8th Edition. Harlow : Pearson Education Limited, 2007. ISBN: 0-321-31379-8.

[12] Spurzem, Bob. 2006. What is Sarbanes-Oxley Act? *CIO Definitions.* [Online] TechTarget, November 21, 2006. [Cited: March 18, 2007.] http://searchcio.techtarget.com/sDefinition/0,,sid19_gci920030,00.html.

[13] The Institute of Electrical and Electronics Engineers. 1998. *IEEE Standard 830-1998.* New York, NY : The Institute of Electrical and Electronics Engineers, Inc., 1998. ISBN: 0-7381-0332-2.

[14] US Federal Trade Commission. The Gramm-Leach Bliley Act. *Privacy Initiatives: Financial Privacy.* [Online] Federal Trade Commission. [Cited: March 17, 2007.] http://www.ftc.gov/privacy/privacyinitiatives/glbact.html.

[15] *Network Effects and Software Development - Implications for Security.* Raman, Jari. 2004. s.l. : IEEE, 2004. 37th Hawaii International Conference on System Sciences. ISBN: 0-7695-2056-1/04.

[16] *Software Engineering for Security: A Roadmap.* Devanbu, Premkumar T. and Stubblebine, Stuart. 2000. Limerick, Ireland : ACM Press, 2000. International Conference on Software Engineering. pp. 227-239. ISBN: 1-58113-253-0.

[17] California State Senate. 2002. SB 1386 Senate Bill - CHARTERED. *California State Senate.* [Online] September 26, 2002. [Cited: March 10, 2007.] http://info.sen.ca.gov/pub/01-02/bill/sen/sb_1351-1400/sb_1386_bill_20020926_chaptered.html.

[18] Abelson, Jenn. 2007. Breach of data at TJX is called the biggest ever. *boston.com: Business.* [Online] The Boston Globe, March 29, 2007. [Cited: April 12, 2007.] http://www.boston.com/business/globe/articles/2007/03/29/breach_of_data_at_tjx_is_called_the_biggest_ever/.

[19] McGregor, J. Patrick. 2007. How the TJX breach may change security awareness. *SC Magazine for IT Security Professional.* [Online] SC Magazine Australia, April 12, 2007. [Cited: April 12, 2007.] http://www.securecomputing.net.au/feature/3417,how-the-tjx-breach-may-change-security-awareness.aspx.

[20] Doggs, Alpha. 2007. Chicago Public Schools latest to fess up to data breach. *Community: Security.* [Online] NetworkWorld, April 7, 2007. [Cited: April 12, 2007.] http://www.networkworld.com/community/?q=node/13573.

[21] Darwin Professional Underwriters. 2007. Tech//404 Data Loss Cost Calculator . *Tech//404 by Darwin.* [Online] Darwin Professional Underwriters, 2007. [Cited: April 12, 2007.] http://www.tech-404.com/calculator.html.

[22] Compuware and Microsoft. 2006. Application Security: How Does your Development Stack Up? *Webcasts On-Demand.* [Online] October 9, 2006. [Cited: April 12, 2007.] http://www.compuware.com/events/forms/app_security.asp?cid=701000000004mADAAY&focus=SecurityChecker&productfocus=SecurityChecker&source=Web+-+Webinar&offering=DevPartner&productfamily=DevPartner&desc=Placement+of+sponsored+webcast+with+Computerworld+onto+Compuware.com.+Speakers%3a+Michael+Leworthy+(Visual+Studio)+%26+Ken+Cowan+(CPWR)&trk=200610-1751.

[23] ISO/IEC. 2004. ISO/IEC 11179, Information Technology -- Metadata Registries (MDR). *ISO/IEC JTC1 SC32 WG2 Development/Maintenance.* [Online] September 15, 2004. [Cited: April 16, 2007.] http://metadata-standards.org/11179/#11179-5. ISO/IEC 11179-1:2004(E).

[24] IEEE. 2007. IEEE Standards Development Online. *IEEE Standards Association.* [Online] IEEE, April 4, 2007. [Cited: April 15, 2007.] http://standards.ieee.org/resources/development/forms/index.html.

[25] National Institute of Standards and Technology. 2006. Information Security Management: A Guide for Managers. *NIST Special Publications.* [Online] NIST, October 2006. [Cited: April 23, 2007.] http://csrc.nist.gov/publications/nistpubs/800-100/SP800-100-Mar07-2007.pdf.

[26] Praxiom Research Group Limited. 2006. ISO 9001 2000. [Online] Praxiom, December 12, 2006. [Cited: April 22, 2007.] http://www.praxiom.com/iso-9001-b.htm.

[27] Praxiom Research Group Limited. 2007. ISO IEC 27001 2005 . [Online] April 7, 2007. [Cited: April 23, 2007.] http://www.praxiom.com/iso-27001.htm.