Forcing Early Binding of Security Using a Design Reference Monitor Concept in Systems Analysis and Design Courses

Corey D. Schou
Idaho State University
schou@mentor.net

Ken Trimmer
Idaho State University
trimkenn@isu.edu

Kevin R. Parker
Idaho State University
parkerkr@acm.org

ABSTRACT

Security information systems are a concern of organizations and governments. However, the topic of security in information systems is addressed lightly in a set of textbooks commonly used in Systems Analysis and Design and Database Design courses. Students will not learn the importance of information security without supplemental materials. At best, information system security will be viewed as a late binding decision in systems design. We propose using the Reference Monitor (RM) as a conceptual framework to introduce security into Systems Analysis and Design courses and subsequent design/ implementation courses.

INTRODUCTION

Information Systems (IS) are ubiquitous and pervasive throughout society, and they form a Critical Information Infrastructure (CII). Information Systems practitioners are becoming concerned with issues such as confidentiality, integrity, and availability of this infrastructure. IS professionals are challenged further by having to ensure privacy and non-repudiation of communications. This complexity can be resolved only by sound analysis and design.

A critical concept is generally lacking from system analysis and conceptual design – the incorporation of security at high or systemic design levels. Limited physical and logical access to early computer information systems developed for mainframe environments commonly considered security issues at the time of physical design or system implementation rather than an early requirement. There is little evidence to indicate that software systems have moved from the reactive 'penetrate-and-patch' mode of the days of yesteryear (Irvine, 1999). With an ever-increasing demand for the availability of data and information, systems security has become a focal point for information technology (IT) expenditures and endeavors. Computer security and information assurance should no longer be late binding events.

A driving factor for computer security, information assurance and the related dimension of confidentiality is a growing body of regulatory legislation. The Gramm-Leach-Bliley (GLB) Act regulates the sharing of personal information about individuals who obtain products or services from financial institutions. Regulations such as the Family Educational Rights and Privacy Act (FERPA) demand high levels of confidentiality and therefore security. In addition to the issue of security is data integrity as represented by the Sarbanes Oxley Act (SOX). SOX imposes organizational and individual penalties for the lack of information integrity in financial reporting. Finally, an underlying principle in the Healthcare Information Portability and Accountability Act (HIPAA), in addition to data privacy and integrity, is the availability of data and resulting information.

Information assurance (IA) and its five main principles – data confidentiality, security, privacy, integrity, and availability – are now expected in a quality information system. To facilitate the incorporation of IA principles in the system analysis and conceptual design phases of systems development projects, we present a perspective that incorporates an abstract model, the reference monitor (RM). We introduce the RM concept as an essential component in embedding high levels of IA in an IS at its conceptual stages. Our goal is to provide a framework that can be included as a component in any course that addresses systems analysis and/or design in an undergraduate curriculum.

To frame this, we first present a discussion of IA, including the concept of the Design Reference Monitor (DRM). Next, we review development methods covered in the classroom by summarizing the methods used in numerous Systems Analysis and Design (SAD) and Database Textbooks. We address the presentation of security and IA in this set of texts. Following this section, we present a discussion of current approaches for addressing security in systems development efforts. We then propose a strategy for introducing IA via the DRM in the undergraduate IS curriculum at the conceptual level. We end our manuscript with a research agenda and general conclusions.

INFORMATION ASSURANCE

Data are observations of the environment while information from a system is that presentation of data that affects ongoing decisions. There are numerous definitions for "information." Very often, information is referred to as the interpretation of data. Thus, the first variance from conventional definitions for the purpose of Information Systems Security (INFOSEC) and IA is that both INFOSEC and IA are measures to protect systems and the information resident in systems.

However, information systems professionals should understand that Information Assurance (IA) is more than computer security or information security. It encompasses the entire lifecycle of data and information from project inception to the demise of the system and the destruction of its contents. Because of the underlying complexity of designing secure systems, security and information assurance are commonly late-binding design functions. When project time constraints are a factor, sometimes they are never bound at all.

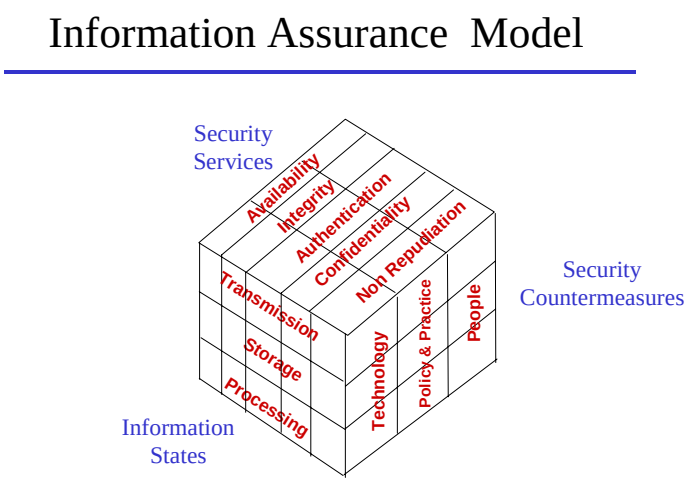
Most information systems professionals recognize that information security protects information systems from unauthorized access to or modification of information in storage, processing, or transit. They may be successful in protecting against denial of service to authorized users and they may build the measures necessary to detect, document, and counter such threats (CNSS, 2003). Notice that this addresses operational systems but it does not address the design perspective.

More recently, information assurance has been defined as operations that protect and defend information and information systems by ensuring their availability, integrity, authentication, confidentiality, and non-repudiation. This includes providing for restoration of information systems by incorporating protection, detection, and reaction capabilities (CNSS, 2003). This definition begins to address some life cycle design issues by introducing the concept of restoration of operations.

Information assurance expands the coverage, responsibilities, and accountability of information systems and security professionals. In addition, it provides a view of information protection as a subset of Information Operations including IA defensive measures. Since Information Operations may include proactive offensive activities, when viewed from this perspective the axiom, "Your offense is only as good as your defense" brings a new perspective to IA and justifies such measures as Active Network Defense.

IA is both multidisciplinary and multidimensional. McCumber asserted this as early as 1991 while developing a model (Figure 1) for computer security (McCumber, 1991). Maconachy et al., (2001) extended this multidimensional model of a robust information assurance program by adding time as an additional factor. The four dimensions of this newer model are represented in figure two, and are:

- Information States
  - Availability, Integrity, Authentication, Confidentiality, and Non-Repudiation
- Security Services
  - Technology, Policy & Procedures and People
- Security Countermeasures
  - Transmission, Storage, and Processing
- Time (On line/offline)

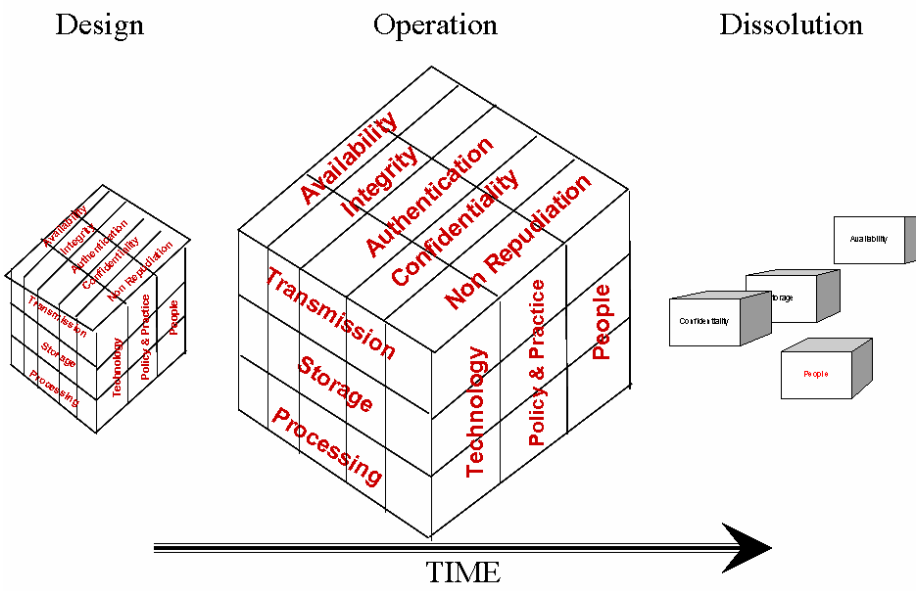


- Figure 1 Information Assurance Model

Although all dimensions are interrelated, from the perspective of the Systems Development Life Cycle, security countermeasures are implementation phase issues as are information states. Although both are dependent upon systems design, information states are dependent on security services, and security countermeasures have direct dependency, through technology, with information states. Our discussion of SAD and IA focuses on the security services dimension. Time, presents a set of issues in Figure 2 to be considered with security services and the other dimensions of the IA model. The IA model demands a life cycle that extends from design, through operations, to the complete dissolution of the system assets.

From a design standpoint, time may be viewed in two ways. First, at any given time access to data may be either accessible on-line or off-line. This introduces the element of risk/exposure to that data via remote unauthorized access means. The most secure system is one that is not connected to any other system. Risk mitigation, as opposed to risk avoidance, takes on a different urgency depending upon connectivity.

The second and more important view of time as it relates to IA is that at any given time the state of our information and information system is in flux. Well-designed systems will include the IA model during all phases of the System Development Life Cycle. During the operational phases, the model is well defined and well implemented. Later in the lifecycle, certain elements of the model may fall away or become less important. In the late stage of a project, one might be most concerned with storage, confidentiality, and availability of the data in the system, while transmission and non-repudiation may become less significant.



• Figure 2 Information Assurance Model Over Time

Time, as a fourth dimension of the integrated model, is not a causal agent of change, but a confounding change agent. As an example, the introduction of new technology, over time requires modifications to other dimensions of the integrated model in order to restore a system to a secure state of operation.

All Security Service dimensions must be introduced early in the design process rather than be applied after the project is complete. There are two dominant design models to be reconsidered with regard to where security is introduced. Both the waterfall and spiral approaches can implement early-binding information assurance as a form of the DRM.

Reference Monitor

Introduced in 1972 by Anderson, the Reference Monitor (RM) is defined in Wikipedia as"… a reference monitor is a secure, always-used, and fully testable module that controls all software access to data objects or devices. The reference monitor verifies the nature of the request against a table of allowable access types for each process on the system". The concept of the RM has been used in graduate coursework at the Naval Post Graduate School as a 'unifying concept' that further enables students to engage in critical thinking regarding embedding security in systems (Irvine, 1999).

The attributes of a software RM are

- Completeness – The RM must be used on every reference of a subject to an object.
- Isolation – The RM and its database must be protected from unauthorized alteration.

- Verifiability – When implemented in code, the RM must be small, well structured, simple, and understandable.

We propose including the concept of the RM not only as a software concept but also as a design tool to be included as a process in systems analysis and design in the educational environment. The next section presents an overview of models commonly presented in a set of widely used systems analysis and design textbooks, and the corresponding presentation of security in the texts.

CURRENT SAD DEVELOPMENT MODELS

Systems Analysis and Design (SAD) is a keystone to building resilient information systems; however, the information assurance components are frequently overlooked in the typical SAD course, as represented by a review of a set of frequently required SAD textbooks in the university curriculum.

Systems Analysis & Design Textbooks

College book representatives of the major publishers were contacted to ascertain which books were their top selling textbooks in systems analysis and design. The goal was to form a list of the best selling textbooks and then review each textbook to determine how security considerations are addressed in each, if at all. The returns from the textbook representatives produced a list of six textbooks:

- Systems Analysis and Design by Dennis and Wixom;
- Modern Systems Analysis and Design by Hoffer, George, and Valacich;
- Systems Analysis and Design by Kendall and Kendall;
- Systems Analysis and Design in a Changing World by Satzinger, Jackson, and Burd;
- Systems Analysis and Design, An Active Approach by Marakas; and
- Systems Analysis and Design Methods by Whitten, Bentley, and Dittman

Our selection was guided by Haworth (2002) and conversations with book representatives at AMCIS 2005. Each of the textbooks in the list was searched for security, authorization, and authentication.

First, we determined the structure of a typical undergraduate SAD text. Our next goal identified models and methods of designing an information system presented in the textbooks. Finally, we wanted to assess the presentation or discussion of security related issues in the set of textbooks and the fit in within the perspective of the textbooks. The intent of this assessment of SAD textbooks is to establish a need for the introduction of the DRM into the SAD process.

Based on our observations, a typical layout for a SAD textbook contains five sections. The first addresses broad issues, including professional systems analysts, project management, problem identification, and other broad, general issues that frequently are presented in multiple courses in a typical undergraduate curriculum. The next section addresses determining system requirements. This topic ranges from one to five subsections, and covers information-gathering techniques and may include presentation of the use case.

The third section of the typical SAD text focuses on modeling techniques. Structured techniques such as process modeling with data flow diagrams (DFD), data models as represented by entity relationship diagrams (ERDs), and the unified modeling language's (UML) class diagrams are typically presented in this section.

The fourth dimension addresses systems design, typically with a discussion of the user interface, inputs and outputs, as well as the physical structure of data and other necessary system components. The typical text will have a section on models that are typically used in focusing requirements and presenting them in a generally understood form. All of the textbooks we assessed, with the exception of a UML text, presented both DFDs and the ERD. In addition to these two structured modeling techniques, the textbooks generally contain a section on object-oriented modeling with UML as represented by the class and other diagrams.

The final section of a typical SAD textbook addresses implementation and emerging topics. This section varies from author to author, and may focus on general issues such as project management and objects, as well as numerous other topics.

Typically, the textbook addresses the SAD effort as either a sequential, structured endeavor, or an iterative approach as represented by the spiral model. We will address these general systems development methods in turn.

Waterfall

Structured methods are frequently represented by the modified waterfall model. In this model software is developed in successive stages (Requirements Definition; System and software Design; Implementation and unit testing; Integration and system testing; Maintenance) with iterations between phases as omissions are discovered. The termination of each phase moves development into the following phase, and as discussed above, this follows the flow of the typical SAD text. This set of processes is commonly referred to as the Systems Development Life Cycle (SDLC) (Marakas, 2006). Dennis and Wixom (2003) note that the waterfall model has two key advantages. System requirements are identified prior to any programming and modifications to the set of requirements are held

at a minimum throughout the SDLC. A drawback for the SDLC as represented by the waterfall model is that, in large-scale projects, requirements that are not identified in earlier stages become increasingly expensive to address in later stages (Marakas, 2006).

Although the waterfall model is often represented as iterating between processes, i.e. requirements can be revisited while in the initial phase of design, etc., other waterfall-type models, such as Parallel Development and Phase Development, do not revisit all SDLC phases (Dennis & Wixom, 2003). A perspective that does address all SDLC phases in an iterative fashion is the spiral model (Satzinger, et al., 2005).

Spiral

The spiral model is an evolution of the waterfall model (and incorporates features of the rapid prototype model). The spiral model represents "an iterative development approach in which each iteration includes a combination of planning, analysis, design, or development steps" (Satzinger, et al., 2005, p 681). Whereas the waterfall model depicts one cycle for each of the four phases, the spiral model allows for quick movement through each phase, resulting in shorter, yet more development cycles. The spiral model emphasizes reiterating earlier stages a number of times as the project progresses. It resembles a series of abbreviated waterfall cycles, each producing an early prototype representing a portion of the entire project. "This approach helps demonstrate a proof of concept early in the cycle, and it more accurately reflects the disorderly, even chaotic evolution of technology" (Kay, 2002, p. 2).

The spiral model consists of a series of steps. System requirements are defined as thoroughly as possible. Detailed information gathering requires interviewing all external or internal users. A preliminary design is created, and then used as a basis for a first prototype of the new system. This is generally a scaled-down version of the system representing an approximation of the characteristics of the final product. A revised prototype is then evolved by:

1) evaluating the first prototype in terms of its strengths, weaknesses, and risks;
2) defining the requirements of the second prototype;
3) planning and designing the second prototype;
4) constructing and testing the second prototype

The customer can make the decision to abort the entire project if the risk is considered too great. Risk factors include projected cost overruns, operating-cost miscalculation, or other factors that could, in the customer's assessment, result in a less-than-satisfactory final product. The revised prototype is evaluated much like the first prototype, and, if necessary, a second refined prototype is developed following the four-step procedure described above. Iterations continue until the customer is satisfied that the refined prototype performs as desired. The final system is constructed based on the accepted prototype, and thoroughly evaluated and tested. Routine maintenance is performed on a continuing basis to prevent failures and minimize downtime (ABC, 2003).

There are two primary approaches to the spiral model. One is a cyclical approach for incrementally developing system requirements and implementation, while the other consists of a set of anchor point milestones designed to ensure stakeholder commitment to feasible and mutually satisfactory system solutions.

Satzinger et al. (2005) present an approach using the spiral model with prototyping. Prototyping, or the use of tools to develop working examples of an information system, focuses on the design and implementation phases of the SDLC (Satzinger et al., 2005).

Security in SAD Textbooks

In general, textbooks mention security in the introduction to systems development and in the design section. Typical textbooks have a few paragraphs or pages on the topic within the entire text. Haworth (2002) identified the lack of security in SAD texts in an examination of a set of texts used in either business programs, or those in computer science. In this manuscript, Haworth proposed categorizing security into three levels as represented by scenarios, and including a person knowledgeable in security as part of the development team.

We updated the presentation and focus on security varies among authors of texts commonly used in business programs, as represented by a recent conference (AMCIS, 2005). To understand the presence of security in them, we examined the reference to 'Security' in the index of the corresponding textbooks. Dennis and Wixom (2003) discuss security in their chapter on Creating the Architecture Design. Whitten et al. (2004), in their textbook's index, reference security with business issues, e-commerce, and logins. In their index and text, Kendall and Kendall (2002) also address security as an e-commence issue. Using each text's index, the Internet and intranet security issues are addressed by Marakas (2006) in his design chapters, as well as access and output controls. Satzinger et al. (2004) initially present security as a requirements issue, yet do not revisit it until their design section. In their UML focused text, Satzinger et al. (2005) present security again as a non-functional requirement, reintroduces a discussion in the design section. At the AMCIS (2005) conference panel discussion on security, one of the SAD textbook authors commented that the next edition of their textbook would contain additional section(s) on security. He estimated that security would be an issue better addressed

by the next edition of the competitor's SAD texts, and is most likely a much larger topic for SAD texts to address throughout future editions (George, 2005).

Security in Database Textbooks

A majority of the texts present security in their design sections. Since database design is part of the overall systems design process, information assurance should also be considered from that perspective. At what point in the database design process do curriculums cover information assurance or security? Note that this does not refer to database security implementation issues like server security and user authorization for specific users at the database administrator level, but rather the how well security issues are incorporated into the overall design approach.

Kroenke (2006), discusses security early from the perspective of security data to define users, groups, and allowed permissions for users and groups. It is vital to consider these issues early in the design process, but the author defers a detailed discussion until a later chapter on multi-user databases. He briefly discusses issues concerning data availability through validation of the data model, but does not present a detailed approach.

Likewise, Rob and Coronel (2004) mention user security and data privacy early in the text, but defer a detailed discussion to a later chapter in Database Administration. A somewhat amorphous approach to data model verification is discussed. Physical security, password security, access rights, audit trails, and data encryption are briefly discussed in chapter 8 on database design issues.

Hoffer, Prescott, and McFadden's (2005) latest edition includes new material on database security policies and technologies, but again it is in a later chapter on database administration. They discuss network level and operating system level security at other points in the text, but in later chapters. Concepts of data model validation are not discussed. They do make mention of security requirements in a diagram of the logical database design phase of the systems development life cycle.

Ricardo (2004) devotes an entire chapter to database security, although the discussion is somewhat belated. A brief, but good discussion of security is included in the introductory chapter. Data model validation is not covered. In spite of the author's emphasis on security, security is not discussed in conjunction with the stages of database design.

Connolly and Begg (1999) also devote an entire chapter to database security issues, but again it comes late in the text. They interweave security through many of the topics throughout the text, but fail to mention it during the database design discussion. Data model validation for information availability is also omitted. Mannino (2004) makes almost no mention of database security with the exception of a section in the database administration chapter. No mention is made of the importance of considering security issues during the design, nor is data model validation discussed.

We believe that the increasing importance of security and the related IA dimensions should be addressed more explicitly at the requirements analysis phase, and introduced into the SAD process in the initial models.

COMMON APPROACHES TO SECURITY

Most software analysis and design efforts pay token respect to security, but do not adequately consider a complete set of IA issues in the design process. Developers are either unable to extract a statement of organizational security policy from those who will own and/or operate the system, or security requirements are simply stated as "security" with no supporting details (Irvine, 1999). When it is discovered that the system lacks critical information assurance capabilities, those features must be added. Universities are producing software developers who do not design with security as an integral part of the process, but instead rely on post-release patches and retrofitting when necessary (Bishop, 2000). Systems that are not designed for security from the ground up can never be repaired in such a way that users are confident of system security (Anderson, 1972).

System security is commonly handled through the penetrate-and-patch approach (Anderson, 1972). Software patches are pieces of software that fix coding errors and minor design problems, and unlike other software releases, such as service packs, they generally do not add new functionality (Silltow, 2005). In the penetrate-and-patch approach someone, generally hackers, security researchers, or security-monitoring companies, discover and exploit a flaw in the code. Vendors then become aware of the flaw and teams develop and release a patch or security solution to deal with the vulnerability. Frequently this turns into a race – patch it before the vulnerability is exploited. Users or IT personnel evaluate the patch, download it, and install it on their systems. Vendor patches often introduce additional flaws that are exploited quickly, and the cycle begins again.

As noted earlier, systems that require frequent repairs do not inspire confidence in the public. Humphrey (1996) points out that no other professional field produces products of uncontrolled quality and then relies on testing to find and fix defects. He notes that with goods like automobiles, consumers intuitively know that when a factory produces a lemon, it will always be a lemon, regardless of the effort spent fixing the defects at the end of the line. The same is often true of software.

The glut of patches causes additional problems as well. Enterprise IT administrators were overwhelmed when fourteen high-profile software vendors released security updates over a two-day period in July 2005 (Naraine, 2005). Numerous patches have made a rigorous patch management process a fundamental security requirement for all of today's organizations (Nicastro, 2003). The patch management process is responsible for such tasks as patch acquisition, impact analysis, vulnerability assessment, deployment, and ongoing patch compliance with policies to insure that systems remain configured correctly. Still, many organizations fear that patches may destabilize existing applications, and as a result, they may delay applying released patches to publicly announced vulnerabilities (Dunagan, et al., 2004).

Since the reactive penetrate-and-patch approach is fraught with problems, it seems logical to be proactive and incorporate security issues in the initial system design. However, security is rarely the most important requirement, and the most secure solution is often not the most desirable (Viega, 2005). Although there are now tools to help address software security, developers still generally do not understand how they should address the problem as a business (Viega, 2005). Developers are unsure where, when, and how these tools should be used (for example, while developing, during daily builds, or during beta), what other kinds of activities should be performed that can't be automated with tools, such as security reviews when originally designing software, and what relative investment should be made in these various activities (Viega, 2005).

Another reason that security is overlooked or minimized is that developer training is insufficient. When many developers learned to write programs in school, they did not receive adequate guidance on how to plan their work or how to produce quality products (Humphrey, 1996). Security is not emphasized in the classroom, and students typically "bang out code of unknown quality and count on compiling and testing to find and fix the defects" (Humphrey, 1996, p. 2). This practice is perpetuated when students enter industry. In other industries, testing may be used to identify design problems, but quality products can only be achieved by using a quality development process. Universities must provide proper education, and quality systems design including IA, must be given top priority.

INCORPORATION INTO CURRICULUM

Secure Code

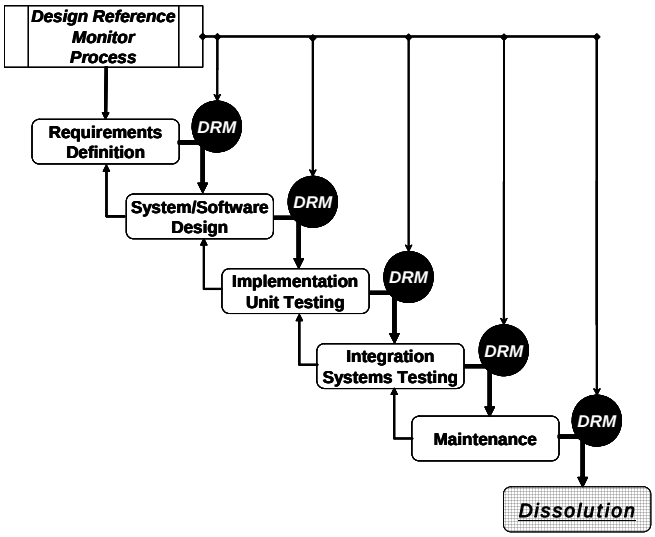
Security coverage is deficient in today's textbooks and curriculum with respect to SAD. In most courses, the concern with functional requirements overshadows other issues and little is being done to emphasize the need to address security requirements (Haworth, 2002). "Indeed, when the space devoted to security is usually only two or three pages in a book of more than 500 pages, one may conclude that the subject is hardly being mentioned" (Haworth, 2002, p. 3). Clearly IS should be a prominent component of an SAD course. Schell, Downey, and Popek (1973) addressed this issue in the early 1970's by noting that computer systems are not secure because security was not a mandatory requirement of the initial design. Nearly thirty years later, Pipkin (2000) laments that it is virtually impossible to add effectively security to a system after it has been designed.

Where then, should IA be covered? The structure of an SAD course should guarantee assurance of information throughout the system life cycle. While awareness of security issues should be interwoven throughout an SAD course, the location of the coverage may vary depending on the predominant model being taught. We propose introducing the concept of the Reference Monitor throughout both Database and SAD courses as a Design Reference Monitor (DRM). The DRM may be defined exactly as the software RM in a design. It must be complete in that it is used between each design step. The rule set it implements must be protected from unauthorized alteration and should be under configuration. The rule set should be parsimonious, well structured, and clear. The first step in any design process should be the specification of the DRM for the proposed system. Over time, the DRM is used during operational maintenance through the dissolution of the system assets. Anything less leads to penetrate and patch mentality.

Waterfall

As noted earlier, in the modified waterfall model software is developed in successive stages with iterations between phases as omissions are discovered. IA must be considered as early as the information-gathering phase, and should play a part in the analysis and design of the new system. Further, IA must be not only requirements-based but also a primary consideration during the testing phase. We propose introducing the concept of verification as used in the RM as a task that must be completed before exiting each phase. As Figure 3 shows, the DRM should be evaluated after each phase is completed and, if it is found to be incomplete, it is modified.

Figure 3 represents the use of the DRM concept within the modified waterfall model. It begins with a process that defines the information assurance needs of a proposed system. This process drives decision making during the requirements definition phase and establishes a baseline DRM. This baseline DRM plans the information assurance requirements and controls from project inception to dissolution of the system assets. As the SDLC continues, the output of each step is reviewed through the DRM. Compliance with the information assurance requirements is confirmed and the DRM is updated as needed. The final role of the DRM in the SDLC is the management of the dissolution of the system that assures proper and planned disposition of assets.

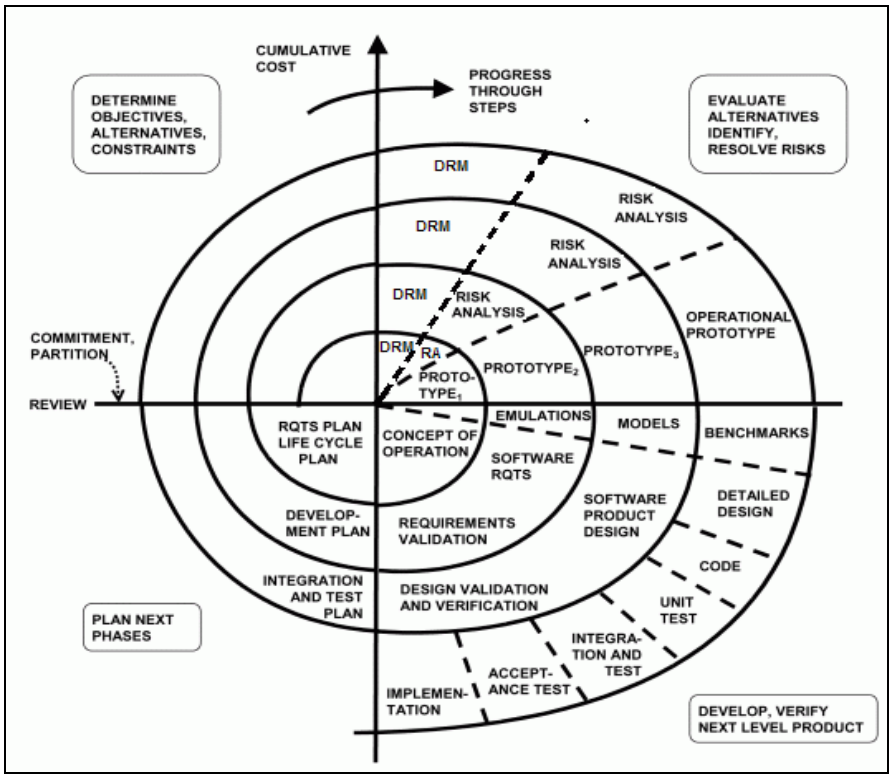


• Figure 3. Modified Waterfall Model with DRM.

This model illustrates the incorporation of the DRM into each phase as defined by the waterfall model. . We shall next present the inclusion of the DRM in the spiral model.

Spiral

In either spiral approach, cyclical or anchor point, the development of a DRM for the projects is critical since requirements are constantly being redefined. In the cyclical approach information assurance needs to be performed at project inception. The initial DRM becomes a component of the initial assessment. In this way, threats to security will be considered as great a risk as cost overruns. It is essential to incorporate IA assessment in each pass around the spiral, as shown in Figure 4, integrating IA awareness into the planning, analysis and design, implementation, and testing phases. Each stage must maintain a relationship with the DRM, and if the DRM is found to be incomplete, it must be modified.



- Figure 4. Modified Spiral Model with DRM.

At the end of the useful life of this type of system, of course, one must include a dissolution process as described earlier.

A modeling strategy addressed by each of the SAD textbooks is process modeling through DFDs. The DFD represents how a set of users or system stakeholders interact with the system. DFDs initially represent an organizational system with a single process, the system. This system accepts inputs from and provides outputs to external entities, generally referred to as the source or sink. The inputs and outputs are data flows.

Utilizing process decomposition, this initial, or context, diagram is then broken down into major system processes, the Level 0 diagram (Hoffer et al., 2005) also referred to as a functional decomposition diagram (Whitten, et al., 2004). This level may contain multiple processes, each corresponding to organizational functions. Each process, in turn, is further decomposed until there is a low enough level of logic. During this decomposition, the process will progressively define inputs and outputs to data stores.

Security should be addressed in initial requirements modeling as represented by the DFD, and we propose inclusion of the DRM as a functional process at Level 0. This will require the analyst to consider security and related IA components early in the development process. Incorporating the DRM into the DFD at this level also implies that all other functional systems are dependent upon the DRM for receiving inputs and sending outputs.

Specifics of the DRM would be decomposed into a Level 1 diagram that would contain, at the minimum, a process to obtain the users specific permission set and a process that generates a view of the system according to permissions. Details of the two sub functions of the DRM would be further decomposed.

By considering the DRM in their initial process models during requirements determination, students will be introduced to the notion of incorporating security and related issues throughout the system. This DRM provides a straightforward, yet powerful concept for the analysis and design of secure information systems.

FUTURE RESEARCH

DRM should be come part of a design configuration control and evaluation model. In addition, it can be implemented in code for systems of systems. It builds information assurance into what Brooks would call a programming systems product. (Brooks, 1995)

The DRM will become a software reference monitor implemented in the form of an avatar. From the Sanskrit for "descent", addressing a deity descended to earth for a specific purpose (Wikipedia). Although one usually thinks of an avatar as a graphic representing a user, in fact we are proposing one as a software instantiation of the designer's intent. The avatar passes down the information assurance intent of the designer to the operational system. It performs a specific purpose and guides the system.

The avatar forces the design to be implemented --- the avatar of the "design god" makes Information Assonance go in before the first design decisions are made all and tracks the process until the dissolution of the system.

CONCLUSIONS

If we are to have sound IA principles applied in the next generation of systems, students must learn that IA must be bound early in the design process. This must be emphasized throughout the curriculum.

We believe that a more detailed focus on issues of security and information assurance is demanded of modern day systems. To conclude our presentation, we provide a set of quotes.

- "Students must understand and appreciate that security must be designed into their products, not added on at the end" (Null, 2004).
- "To ensure safe computing, the security (and other desirable properties) must be designed in from the start. To do that, we need to be sure all of our students understand the many concerns of security, privacy, integrity, and reliability" (Spafford, 1997).
- "By moving to an educational system that cultivates an appropriate knowledge of security, we can increase the likelihood that our next generation of IT workers will have the background needed to design and develop systems that are engineered to be reliable and secure" (Irvine, Chin, and Frincke, 1998).

The academic community must change the way it leads students into the profession. The need is immediate and apparent, and critical.

REFERENCES

1. Akash Bharti Computech (ABC). (2003). Spiral model. Retrieved on August 7, 2005, from http://funnyprofessor.com/softeng/softeng4.html
2. AMCIS 2005. Americas Conference on Information Systems, August 8 – 11, 2005, Omaha, Nebraska, USA

3. Anderson, J. P. (1972). *Computer Security Technology Planning Study*. Technical Report ESD-TR-73-51, Air Force Electronic Systems Division, Hanscom AFB, Bedford, MA.
4. Bishop, M. (2000). Education in Information Security. *IEEE Concurrency*, Oct.-Dec., pp. 4-8.
5. Brooks, Fred. (1995) *The Mythical Man-Month: Essays on Software Engineering, 20th Anniversary Edition*, Reading, MA: Addison-Wesley.
6. Committee for National Security Systems (CNSS). (2003). National Information Systems Security Glossary. CNSSI 4009, Fort Meade, MD. Sept. Retrieved on August 8, 2005, from http://www.cnss.gov/instructions.html
7. Connolly, T.M. and Begg, C.E. (2005). *DataBase Systems: A Practical Approach to Design, Implementation, and Management* (4th ed.), Reading, MA: Addison-Wesley.
8. Dennis, A. and Wixom, B. (2003). *Systems Analysis and Design* (2nd ed.), New York, NY: John Wiley & Sons, Inc.
9. Dunagan, J., Roussev, R., Daniels, B., Johnson, A., Verbowski, C., and Wang, Y. (2004). Towards a self-managing software patching process, using black-box persistent-state manifests. *Proceedings of the International Conference on Autonomic Computing* (ICAC'04), pp. 106–113, New York, NY, May.
10. George, J. F. (2005). Are Prevailing Theories and Practices of IS Security Management Adequate? An Evaluation and Call-to action, *Panel at AMCIS 2005.*
11. Haworth, D. (2002). *Security Scenarios in Analysis and Design*. The SANS Institute.
12. Hoffer, J.A., George, J.F. and Valacich, J.S. (2004). *Modern Systems Analysis & Design* (4th ed.), Upper Saddle River, NJ: Prentice-Hall.
13. Hoffer, J.A., Prescott, M.B., and McFadden, F.R. (2005). *Modern Database Management* (7th ed.), Upper Saddle River, NJ: Prentice-Hall.
14. Humphrey, W.S. (1996). Making Software Manageable. *Crosstalk*, STSC, Hill Air Force Base, Utah, December, pp. 3-6.
15. Irvine, C. E. (1999). The Reference Monitor Concept as a Unifying Principle in Computer Security Education. *Proceedings of the IFIP TC11 WG 11.8* First World Conference on Information Security Education, pp. 27-37, Kista, Sweden, June.
16. Irvine, C. Chin, S., and Frincke, D. (1998). Integrating Security into the Curriculum. IEEE Computer, 31 (12), pp. 25-30.
17. Kay, R. (2002). QuickStudy: System Development Life Cycle. *ComputerWorld*, May 14, Retrieved on August 8, 2005, from http://www.computerworld.com/printthis/2002/0,4814,71151,00.html
18. Kendall, K.E. and Kendall, J.E. (2002). *Systems Analysis and Design* (5th ed.), Upper Saddle River, NJ: Prentice-Hall.
19. Kroenke, D.V. (2006) *Database Processing: Fundamentals, Design, and Implementation* (9th, ed.), Upper Saddle River, NJ: Prentice-Hall.
20. Maconachy, W.V., Schou, C., Ragsdale, D., and Welch, D. (2001). A Model for Information Assurance: An Integrated Approach. Proceedings of the 2001 IEEE Workshop on Information Assurance and Security, pp. 306-310, United States Military Academy, West Point, NY, June 5-6.
21. Mannino, M.V. (2004). *Database Design, Application Development, and Administration* (2nd ed.), Boston, MA: McGraw-Hill/Irwin.
22. Marakas, G.M. (2006). *Systems Analysis and Design, An Active Approach* (2nd ed.), Boston, MA: McGraw-Hill/Irwin.
23. McCumber, J. (1991). Information Systems Security: A Comprehensive Model. Proceedings of the 14th National Computer Security Conference. National Institute of Standards and Technology. Baltimore, MD. October.
24. Naraine, R. (2005). Security Patch Deluge: A Double-Edged Sword. EWeek.com, July 14, Retrieved on August 7, 2005, from http://www.eweek.com/article2/0,1759,1837120,00.asp
25. Nicastro, F.M. (2003). Security Patch Management, INS Whitepaper, Retrieved on August 7, 2005, from http://www.ins.com/downloads/whitepapers/ins_white_paper_security_patch_mgmt_ 0303.pdf
26. Null, L. (2004). Integrating Security Across the Computer Science Curriculum. *Journal of Computing Sciences in Colleges*, 19 (5), p.170-178.

27. Pipkin, D. (2000). *Information Security: Protecting the Global Enterprise*. Upper Saddle River, NJ: Prentice Hall.

28. Ricardo, C. (2004). *Databases Illuminated*, Sudbury, MA: Jones & Bartlett Publishers.

29. Rob, P. and Coronel, C. (2004). *Database Systems: Design, Implementation and Management* (6th ed.), Boston, MA: Course Technology.

30. Satzinger, J.W., Jackson, R.B., and Burd, S.D. (2004). *Systems Analysis & Design in a Changing World*, Boston, MA: Thomson/Course Technology.

31. Satzinger, J.W., Jackson, R.B., and Burd, S.D. (2005). *Object-Oriented Analysis & Design with the Unified Process*, Boston, MA: Thomson/Course Technology.

32. Schell, R.R., Downey, P.J., and Popek, G.J. (1973). Preliminary Notes on the Design of Secure Military Computer Systems, MCI-73-1, The MITRE Corporation, Bedford, MA 01730 (Jan.). Retrieved on August 11, 2005, from http://seclab.cs.ucdavis.edu/projects/history/CD /index.html#sche73

33. Silltow, J. (2005). Getting Patches Under Control. IT Audit, Vol. 8, February 15, Retrieved on August 9, 2005, from http://www.theiia.org/itaudit/index.cfm?fuseaction=forum&fid=5592

34. Spafford, E.H. (1997). One view of a critical national need: Support for information security education and research. August 11, 2005, from http://www.fas.org/irp/congress/1997_hr/h970211s.htm

35. Viega, J. (2005). Security: Problem Solved? ACM Queue, 3 (5), pp. 40-50, Retrieved on August 9, 2005, from http://www.acmqueue.org/modules.php?name=Content&pa=showpage&pid=311 &page=1

36. Whitten, J.L., Bentley, L.D., and Dittman, K.C. (2004). *Systems Analysis and Design Methods* (6th ed.), Boston, MA: McGraw-Hill/Irwin:

37. Wikipedia (2005) WWW.Wikipedia.Org