ENFORCING EARLY IMPLEMENTATION OF INFORMATION ASSURANCE PRECEPTS THROUGHOUT THE DESIGN PHASE

Ken Trimmer
College of Business
Idaho State University
trimkenn@isu.edu

Corey Schou
College of Business
Idaho State University

Kevin Parker
College of Business
Idaho State University

ABSTRACT

Secure information systems are of great concern to organizations and governments. However, the topic of information systems security is inadequately addressed in most textbooks commonly used in Systems Analysis and Design and Database Design courses. Students do not learn the importance of information security unless supplemental materials are provided. At best, information system security is often viewed as a broad non-functional requirement and 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 Database Design courses as well as subsequent design/ implementation courses.

Keywords: systems development life cycle, requirements analysis phase, design reference monitor, information assurance, McCumber, MSR model

I. INTRODUCTION

Information systems (IS) are ubiquitous and pervasive throughout society, and they are part of a critical information infrastructure (CII). A critical information infrastructure is a communications or information service whose availability, reliability and resilience is essential to the functioning of a modern economy, security, and other essential social values [Cukier, 2005]. Information systems practitioners must

be concerned with issues such as confidentiality, integrity, and availability of any such infrastructure. Information Systems professionals are challenged further by having to ensure privacy and non-repudiation of communications. Earlier research established that the complexity can be resolved only by sound analysis and design [Schou et al., 2005].

The incorporation of security at high or systemic design levels is generally lacking in system analysis and conceptual design. In early mainframe information systems limited physical and logical access permitted security issues to be postponed until physical design or system implementation. This practice, referred to as late binding, involves the completion of a significant portion of a design without binding design components to a particular kind of implementation. There is little evidence to indicate that software systems have moved from yesteryear's reactive 'penetrate-and-patch' mode [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. Designers no longer have the luxury of deferring computer security and Information Assurance to late binding.

A driving factor for computer security, Information Assurance and the embedded dimension of confidentiality is a growing body of regulatory legislation in the United States. 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 responsibility and 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 security services – data confidentiality,

authentication, non-repudiation, integrity, and availability – are now expected attributes of all quality information systems. To facilitate the incorporation of Information Assurance 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) and combines it with the programming concept of binding. We introduce the RM concept as an essential component in embedding high levels of Information Assurance in an Information System 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 a curriculum.

BINDING

The concept of binding originally referred to the point in time at which variables and expressions in a program are bound to their values and assigned a data type. With early, or static, binding this is done at compilation time, i.e, before the program is run. In late, or dynamic, binding the routine or object is linked at runtime based on the conditions at that moment. The authors propose adapting the concept from programming and applying it to the design phase. Software designers should be encouraged to early-bind security by ensuring that it is considered at each design phase. Security should be bound to entities such as objects and functions before the first line of code is attempted. We further propose using the Reference Monitor concept as a design-time tool.

REFERENCE MONITOR

Introduced in the early 1970s Anderson [1972], the Reference Monitor (RM) is software that validates all references to programs or data (objects) according to an access authority. It should be a fully testable subsystem that controls all software access to data objects or devices. 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 criteria for making security decisions are maintained

within the operating system, and are implemented via the Reference Monitor. The Reference Monitor is software that is interposed between all subjects and objects within the system. The Reference Monitor acts as an intermediary and regulates how subjects can access objects within the environment. A Reference Monitor first automatically logs a request for access and then provides explicit authorization for access if the request can be validated against a self-contained database of stored criteria. The Reference Monitor is always invoked, tamper-proof, and small/verifiable.

The attributes of a software Reference Monitor are:

- Completeness – The RM must be used on every reference of a subject to an object – always invoked.
- Isolation – The RM and its database must be protected from unauthorized alteration – tamper proof.
- Verifiability – When implemented in code, the RM must be small, well structured, simple, and understandable – testable.

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. This adaptation of the programming concept would encourage the integration of Information Assurance throughout the design phase.

The use of the Design Reference Monitor derived from this concept is discussed later and shown in figures 3 and 4.

EXAMPLE

The use of these precepts in assurance of the design of a medical/healthcare information system would include developing a complete analysis of the HIPAA requirements, SOX compliance, organizational security requirements, local privacy laws, programming and design standards, etc. These would be compiled to establish a complete preliminary assurance baseline. This document should be developed without reference to other design requirements and should be externally reviewed for completeness, applicability, and accuracy. The document should be formally

accepted by management early in the design and development process. This document forms the core of the Design Reference Monitor.

As the design and development process continues, the proposals are checked against the Reference Monitor. If deviations are identified, senior management must either formally accept the change in risk or justify a change to the baseline assurance requirements. If the baseline is changed, this change is reflected in all future steps. This is done at every stage in the software development process, and permeates every phase of the development life cycle.

HEADING HERE?

To frame this, we first present a discussion of Information Assurance, including the introduction of the Design Reference Monitor (DRM) as part of the design process. The DRM should define the security rule set at every stage. For example, “every object in the database will have access controls” or “all program components will check for potentials for buffer overflow” or “all software will be tested for security functions in addition to operational testing” [Schou and Shoemaker 2007]. If each appropriate DRM function is bound to every design phase, security will be easier to establish. We recommend that Information Assurance should be a design-time activity that is bound early to the process. Next, we review development methods covered in the classroom by summarizing the methods used in numerous traditional or structured Systems Analysis and Design (SAD) textbooks. We address the presentation of security and Information Assurance within 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 Information Assurance via the DRM in the Information Systems curriculum at the conceptual level. Finally, we present a research agenda and general conclusions.

II. INFORMATION ASSURANCE

Data are observations of the environment, while information generated by a system is the 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 Information Assurance is that both INFOSEC and Information Assurance are measures to protect systems and the information (rather than data) resident in those systems.

Information systems professionals should understand that Information Assurance transcends simple 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 inherent complexity in 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.

Most information systems professionals consider information security from an operational perspective but overlook the design perspective. They are aware of their responsibility to protect 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]. However the incorporation of Information Assurance in the design stages can make operational security more manageable.

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 Information Assurance defensive measures.

Information Assurance 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 including time as an additional factor (this is also known as the MSR model – [ACM 2005]). The four dimensions of this newer model are represented in Figure 2, and are:

- Information States
  - Availability, Integrity, Authentication, Confidentiality, and Non-Repudiation
- Security Services
  - Technology, Policy and Procedures, and People
- Security Countermeasures
  - Transmission, Storage, and Processing
- Time (From inception to dissolution)

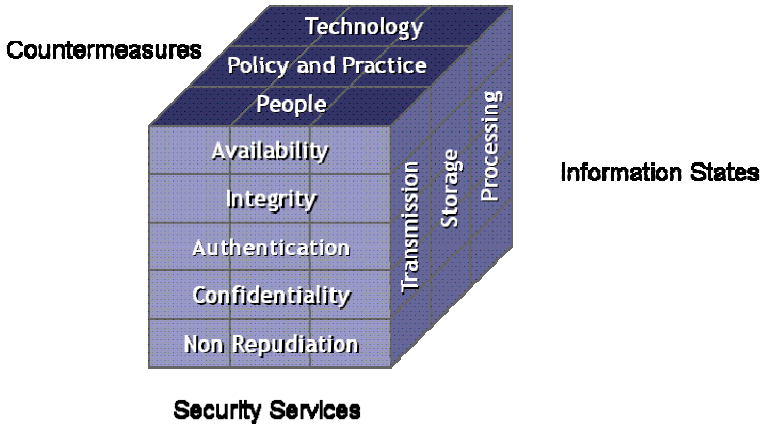

Figure 1. Information Assurance Model

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, on information states. Our discussion of Systems Analysis and Design and Information Assurance focuses on the security services dimension in Figure 1. Time presents another set of issues in Figure 2 to be considered with security services and with the other dimensions of the Information Assurance model. The Information Assurance 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 Information Assurance is that at any given time the state of our information and information system is in flux. Well-designed systems will involve the Information Assurance model through all phases of the System Development Life Cycle. During the operational phases, the model is well defined and well implemented. Later in the life cycle, 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 as they are already embedded within the system.

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

structured design models to be reconsidered with regard to where security is introduced. Both the waterfall and spiral approaches can incorporate early-binding Information Assurance through the DRM.

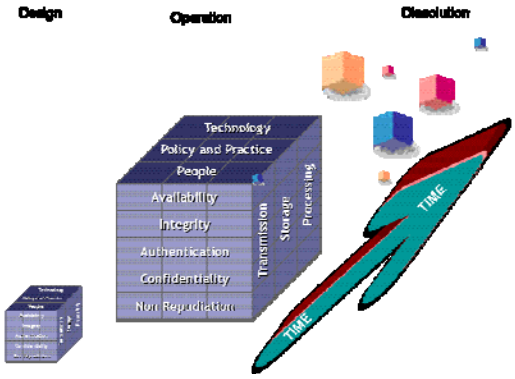

Figure 2. Information Assurance Model Over Time

III. CURRENT SYSTEMS ANALYSIS AND DESIGN DEVELOPMENT MODELS

This section presents an overview of models commonly presented in a set of widely used, traditional Systems Analysis and Design textbooks and the corresponding presentation of security in the texts.

Systems Analysis and Design (SAD) is a keystone to building resilient information systems; however, the Information Assurance components are frequently overlooked in the typical Systems Analysis and Design course, as represented by a review of a set of frequently required Systems Analysis and Design textbooks in the university curriculum. We were examined them for statements such as “System security policy not only establish feasibility but be a requirement of all systems. Building the policy *a priori* must be mandatory.”

The same observations can be made regarding texts that discuss database design concepts. Our textbook analysis follows the approach used by Haworth [2002].

SYSTEMS ANALYSIS AND DESIGN TEXTBOOKS

Representatives of the major publishers of college texts 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 – Dennis and Wixom, 2003; Hoffer and Valacich, 2005; Kendall and Kendall 2002; Satzinger and Burd , 2004; Marakas; 2006; Whitten and Dittman, 2004 – each of which was searched for coverage of security, authorization, and authentication:

First, we determined the structure of a typical undergraduate Systems Analysis and Design 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 their fit within the perspective of the textbooks. The intent of this assessment of Systems Analysis and Design textbooks is to establish a need for the introduction of the DRM into the Systems Analysis and Design process.

Based on our observations, a typical structure for a Systems Analysis and Design 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 the determination of 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 Systems Analysis and Design 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 has 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 Systems Analysis and Design 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 Systems Analysis and Design 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 is the same flow followed by the typical Systems Analysis and Design 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 to a minimum throughout the SDLC. A drawback of 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 and 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].

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 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.

At any time, 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.

SECURITY IN SYSTEMS ANALYSIS AND DESIGN 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] identifies the lack of security in Systems Analysis and Design texts in an examination of a set of texts used in either business programs, or those in computer science. Haworth proposes categorizing security into three levels as represented by scenarios, and including a person knowledgeable in security as part of the development team.

The presentation and focus on security varies among authors of texts commonly used in business programs. To understand the presence of security in textbooks we examined the reference to ‘Security’ in the index of each text. Dennis

and Wixom [2003] discuss security in their chapter on Architecture Design. Whitten et al. [2004] reference security with business issues, e-commerce, and logins. Kendall and Kendall [2002] address security as an e-commence issue. 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, and reintroduce the discussion in the design section. At the AMC Information Systems [2005] conference panel discussion on security, one of the Systems Analysis and Design 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 Systems Analysis and Design texts, and is most likely a much larger topic for Systems Analysis and Design texts to address throughout future editions [George, 2005].

SECURITY IN DATABASE TEXTBOOKS

A majority of database 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. Authors like Kroenke [2006] include an early discussion of security from the perspective of security data to define users, groups, and allowed permissions for users and groups; however, as Schou [2006] points out, most texts omit thorough discussions.

IV. COMMON APPROACHES TO SECURITY

Most software analysis and design efforts pay token respect to security, but

do not adequately consider a complete set of Information Assurance 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 eventually discovered that the system lacks critical Information Assurance capabilities, those features must be added. Universities are producing software developers who do not consider security to be an integral part of the design 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 modern security-conscious 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 ensure 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 incorporate proactively 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].

Frequently security is overlooked or minimized since designer and 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 should feel obliged to provide proper education, and quality systems design including Information Assurance must be given top priority.

V. INCORPORATION INTO CURRICULUM

SECURE CODE

Security coverage is deficient in today's textbooks and curriculum with respect to Systems Analysis and Design. 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 Information Systems should be a prominent component of a Systems Analysis and Design course. Schell, Downey, and Popek [1973] addressed this issue in the early 1970's by noting that computer systems are not secure because security is not a mandatory requirement of the initial design. Nearly thirty years later, Pipkin [2000] laments that it is virtually impossible to add effective security to a system after it has been designed.

Where then, should Information Assurance be covered? The structure of a Systems Analysis and Design course should guarantee assurance of information throughout the system life cycle. While awareness of security issues should be interwoven throughout a Systems Analysis and Design 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 Systems Analysis and Design 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 the penetrate-and-patch mentality [Schou et al., 2005].

DESIGN MODELS

Design establishes how a system will function. There are two fundamental forms of design models – waterfall and spiral. Each has its inherent advantages. Traditionally the waterfall introduces security one of two places – at the beginning and it is never re-visited, or at the end as an afterthought. The spiral approach allows for reconsideration of the binding decisions after each pass through the cycle, while the waterfall requires that management check the Information Assurance process between major steps.

Design is not a monolithic activity. Developing secure systems requires that they be built according to an assurance plan that is documented in the Design Reference Monitor, established and modified in accordance with plan.

Waterfall

As noted above, in the modified waterfall model software is developed in successive stages with iterations between phases as omissions are discovered. Information Assurance must be considered as early as the information-gathering phase, and should play a part in the analysis and design of the new system. Information Assurance 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 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.

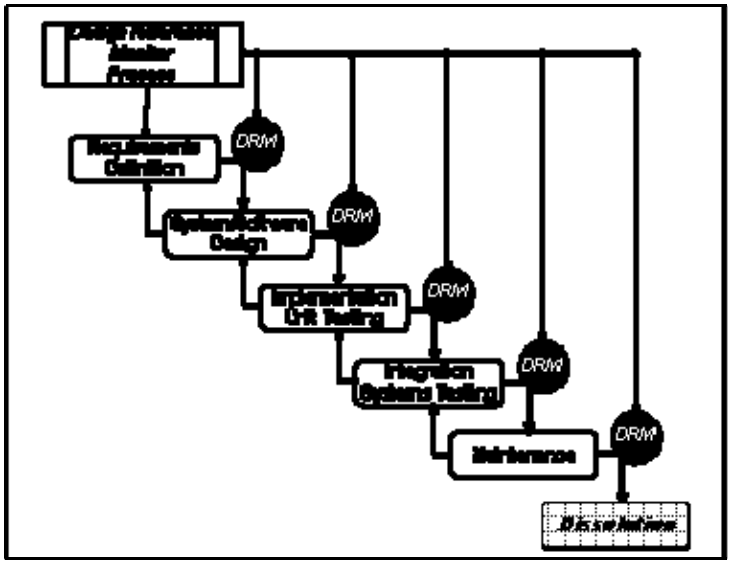

Figure 3. - Modified Waterfall Model with DRM

This model illustrates the incorporation of the DRM into each phase as defined by the waterfall model. We 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 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 Information Assurance assessment in each pass around the spiral, as

shown in Figure 4, integrating Information Assurance 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. 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 Systems Analysis and Design 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.

Using 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 Information Assurance 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.

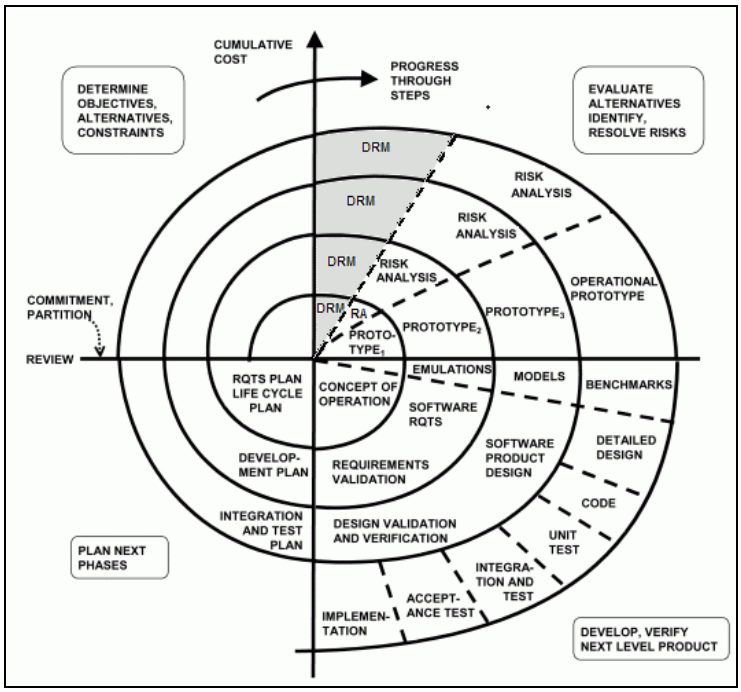

Figure 4. Modified Spiral Model with DRM

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.

VI. FUTURE RESEARCH

The DRM should become part of a design configuration control and evaluation model. In addition, it can be implemented in code for systems of systems and it builds Information Assurance into what Brooks would call a programming systems product. [Brooks, 1995].

Several natural extensions of this model are suggested:

- Examine database texts since preliminary inspections indicate that the late binding phenomenon is also seen in database texts.
- Extend the analysis curriculum issue regarding newer versions of texts and inter-rater reliability.

- Build the technical design extension to tie to programming curriculum.
- Develop the model for implementation as part of the programming and design structure (avatar).

The DRM will become a software Reference Monitor implemented in the form of an avatar. The word "avatar" is derived from the Sanskrit word for "descent", addressing a deity descended to earth for a specific purpose [Wikipedia, 2005]. Although one usually thinks of an avatar as a graphic that represents a user, 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.

VII. CONCLUSIONS

If we are to have sound Information Assurance principles applied in the next generation of systems, students must learn that Information Assurance must be bound early in the design process. This must be emphasized throughout the curriculum. Currently, students represent data elements predominantly through the database model as reflected in either ERDs or Class Diagrams. Data states are represented in the Class Diagram and the association between entities and stores in the ERD and DFD respectively. Future students must learn to incorporate the Information Assurance principles embodied by the Design Reference Monitor into their data model as generated by either structured or object oriented methods during the design phases of systems development.

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, apparent, and critical. Ultimately, Information Assurance cannot be outsourced or added after the fact.

VIII. REFERENCES

(2003) "Spiral Model," *Akash Bharti Computech (ABC),* http://funnyprofessor.com/softeng/softeng4.html (current Aug. 7, 2005).

ACM 2005 Computing Curricula for Information Technology Volume, http://www.acm.org/education/curric_vols/IT_October_2005.pdf (current Oct. 2005).

AMCInformation Systems2005, Americas Conference on Information Systems, August 8 – 11, 2005, Omaha, Nebraska.

Anderson, J.P. (1972) "Computer Security Technology Planning Study," *Technical Report ESD-TR-73-51*, Hanscom AFB, Bedford, MA: Air Force Electronic Systems Division.

Bishop, M. (2000) "Education in Information Security," *IEEE Concurrency*, Oct.-Dec., pp. 4-8.

Brooks, F. (1995) *The Mythical Man-Month: Essays on Software Engineering, 20th Anniversary Edition,* Reading, MA: Addison-Wesley.

Committee for National Security Systems (CNSS) (2003) *National Information Systems Security Glossary*, CNSSI 4009, Fort Meade, MD. Sept. http://www.cnss.gov/instructions.html (current Aug. 8, 2005).

Connolly, T.M. and C.E. Begg (2005) *DataBase Systems: A Practical Approach to Design*, *Implementation, and Management, 4th edition*, Reading, MA: Addison-Wesley.

Cukier, K. (2005) "Critical Information Infrastructure Protection," *A Report of the 2005 Rueschlikon Conference on Information Policy*.

Dennis, A. and B. Wixom (2003) *Systems Analysis and Design, 2nd edition,* New York, NY: John Wiley and Sons, Inc.

Dunagan, J., et al. (2004) "Towards a Self-Managing Software Patching Process, Using Black-Box Persistent-State Manifests," *Proceedings of the International Conference on Autonomic Computing* (ICAC'04), New York, NY, May, pp. 106–113.

George, J.F. (2005) "Are Prevailing Theories and Practices of Information SystemsSecurity Management Adequate? An Evaluation and Call-to Action," *Panel at AMCInformation Systems2005.*

Haworth, D. (2002) *Security Scenarios in Analysis and Design,* The SANS Institute.

Hoffer, J.A., J.F. George, and J.S. Valacich (2004) *Modern Systems Analysis and Design, 4th edition,* Upper Saddle River, NJ: Prentice-Hall.

Hoffer, J.A., M.B. Prescott, and F.R. McFadden (2005) *Modern Database Management, 7th edition*, Upper Saddle River, NJ: Prentice-Hall.

Humphrey, W.S. (1996) "Making Software Manageable," *Crosstalk*, STSC, Hill Air Force Base, Utah, December, pp. 3-6.

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, Kista, Sweden, June, pp. 27-37.

Irvine, C., S. Chin, and D. Frincke (1998) "Integrating Security into the Curriculum," *IEEE Computer* (31)12, pp. 25-30.

Kay, R. (2002) "QuickStudy: System Development Life Cycle," *ComputerWorld*, May 14, http://www.computerworld.com/printthis/2002/0,4814,71151,00.html (current Aug. 8, 2005).

Kendall, K.E. and J.E. Kendall (2002) *Systems Analysis and Design, 5th edition*, Upper Saddle River, NJ: Prentice-Hall.

Kroenke, D.V. (2006) *Database Processing: Fundamentals, Design, and Implementation*, *9th edition*, Upper Saddle River, NJ: Prentice-Hall.

Maconachy, W.V. et al. (2001) "A Model for Information Assurance: An Integrated Approach," *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*, United States Military Academy, West Point, NY, June 5-6, pp. 306-310.

Mannino, M.V. (2004) *Database Design, Application Development, and Administration, 2nd edition*, Boston, MA: McGraw-Hill/Irwin.

Marakas, G.M. (2006) *Systems Analysis and Design, An Active Approach, 2nd edition*, Boston, MA: McGraw-Hill/Irwin.

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.

Naraine, R. (2005) “Security Patch Deluge: A Double-Edged Sword,” EWeek.com, July 14, http://www.eweek.com/article2/0,1759,1837120,00.asp (current Aug. 7, 2005).

Nicastro, F.M. (2003) “Security Patch Management,” INS Whitepaper, http://www.ins.com/downloads/whitepapers/ins_white_paper_security_patch_mgmt_0303.pdf (current Aug. 7, 2005).

Null, L. (2004) “Integrating Security Across the Computer Science Curriculum,” *Journal of Computing Sciences in Colleges* (19)5, pp.170-178.

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

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

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

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

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

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

Schou, C.K. K. Trimmer, and K.R. Parker (2005) "The Design Reference Monitor Concept in Systems Analysis and Database Design Courses," *Proceedings of the International Conference on Informatics Education and Research*, Las Vegas, Nevada, December 9-11, pp. 321-331.

Schou, C. and D. Shoemaker (2007) *Information Assurance for the Enterprise A Roadmap to Information Security*, New York, NY: McGraw Hill ISBN: 0072255242.

Silltow, J. (2005) “Getting Patches Under Control,” *IT Audit* (8) February 15, http://www.theiia.org/itaudit/index.cfm?fuseaction=forum&fid=5592 (current Aug. 9, 2005).

Spafford, E.H. (1997) “One View of a Critical National Need: Support for Information Security Education and Research,” http://www.fas.org/irp/congress/1997_hr/h970211s.htm (current Aug. 11, 2005).

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

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

Wikipedia (2005) http://www.wikipedia.org

ABOUT THE AUTHORS

Ken Trimmer is an Associate Professor of Computer Information Systems in the College of Business at Idaho State University. He holds a PhD. in Information Systems from the University of South Florida. His pedagogical research focuses on educational issues in systems analysis and design. Other current research interests focus on adoption and implementation issues with enterprise systems in academic institutions and healthcare. Dr. Trimmer has numerous publications in journals and conference proceedings, and reviews for both journals and conferences.

Corey Schou is a University Professor of Informatics and Associate Dean of Information Systems in the College of Business at Idaho State University. He holds a PhD from Florida State University. Dr. Schou is the director of the Informatics Research Institute and directs both NIATEC (National Information Assurance Teaching and Education Curriculum), part of Idaho State University's National Center of Excellence for Information Assurance and the Simplot Decision Center. He has published numerous journal articles on Information Assurance and has focused on educational standards for this subject. Schou currently serves as the Chair of the Colloquium for Information Systems Security Education.

Kevin Parker is a Professor of Information Systems in the College of Business at Idaho State University. He holds a PhD. in Information Systems from Texas Tech University. Dr. Parker is an active participant in the research community, serving as editor of a journal and reviewing for numerous journals and conferences. His excellence in teaching and research has been recognized by the College of Business as an outstanding teacher and researcher.