"Information Systems Security Design Methods: Implications for Information Systems Development"

Article *in* ACM Computing Surveys · December 1993

DOI: 10.1145/162124.162127 · Source: DBLP

1 author:

Richard Baskerville
Georgia State University
223 PUBLICATIONS **17,825** CITATIONS

SEE PROFILE

Information Systems Security Design Methods: Implications for Information Systems Development

RICHARD BASKERVILLE

School of Management, Binghamton University, Binghamton, New York 13901

The security of information systems is a serious issue because computer abuse is increasing. It is important, therefore, that systems analysts and designers develop expertise in methods for specifying information systems security. The characteristics found in three generations of general information system design methods provide a framework for comparing and understanding current security design methods. These methods include approaches that use checklists of controls, divide functional requirements into engineering partitions, and create abstract models of both the problem and the solution. Comparisons and contrasts reveal that advances in security methods lag behind advances in general systems development methods. This analysis also reveals that more general methods fail to consider security specifications rigorously.

Categories and Subject Descriptors: C.0 [**Computer Systems Organization**]: General—*systems specification methodology*; H.1.1 [**Information Systems**]: Systems and Information Theory—*value of information*; H.1.2 [**Information Systems**]: User/Machine Systems—*human factors*; K.6.1 [**Management of Computing and Information Systems**]: Project and People Management—*systems and analysis and design*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*authentication; insurance; invasive software; physical security*

General Terms: Management, Security

Additional Key Words and Phrases: Checklists, control, integrity, risk analysis, safety, structured systems analysis and design, system modeling

INTRODUCTION

Most designers do not intentionally set out to design an information system that is unsafe or insecure. Yet research into specification methods for the analysis and design of information systems security often seems esoteric and remote. The purpose of this article, therefore, is to provide an analytical description of the evolution of information systems (IS) security analysis and design methods. To make this description interesting to a broad audience, we will approach these methods by comparing them with more general information systems development methods. In this way, we can understand the current techniques for creating safe computing resources and recognize critical avenues for new research in general systems development methods.

Importance of Security

The gravity of the systems security issue is reflected in widely known studies of computer abuse [Parker 1976; Whiteside 1978; BloomBecker 1990], computer viruses [Fites et al. 1989; Hruska 1990], and illegitimate computer hacking or cracking [Landreth 1989; Stoll 1989;

CONTENTS

Hafner and Markoff 1991]. A study by the American Bar Association of 283 large companies found that 48 percent of these companies considered themselves victims of computer crime over a 12-month period [American Bar Association 1984]. Hoffer and Straub [1989], reporting a survey of U.S. managers and systems professionals, found that one in five organizations experiences one or more security breaches in a three-year period. They suggest that this is a fraction of actual abuse because often such breaches are concealed when discovered. Organizations wish to avoid the publicity that accompanies computer abuse. Hoffer and Straub concede that most of these events may go undetected anyway.

The growing concern for computer abuse is global. For example, Dixon et al. [1992] report a London Business School study that estimated 12 percent of computer-reliant companies were victims of fraud at an average loss of £46,000 (approximately $75,000). They also present an analysis of U.K. Audit Commission surveys that indicate a rise in the number of computer fraud incidents from 67 in 1981 to 118 in 1987 with the average loss per incident rising from £31,000 ($50,000) in 1981 to £262,000 ($425,000) in 1987. The French information technology security society (Clusif) reported a 15.5 percent increase in losses caused by security breaches in 1991 with a total estimated cost of 10.4 billion francs ($2 billion) [Gannon 1992]. Solarz [1987] analyzed embezzlement in Sweden between 1981 and 1983. One out of ten of the serious cases was computer related, each of these incidents averaging SEK196,000 ($30,000). Japan also has a rising number of computer crimes; in particular, the Japanese witnessed one of the most startling increases in automated bank teller abuse in the early 1980s [Computerworld 1983].

More recently, spectacular abuse from malicious programs has supplanted computer crime as the primary security concern. The Internet worm crippled one of the world's largest and most complex computer networks in a matter of hours [Spafford 1989]. In 1992, threat of the Michelangelo virus became an international media event. While the effects of the virus were not catastrophic, substantial disruption occurred in microcomputer service *world-wide* while extensive preventive measures were undertaken by cautious system operators [Highland 1992]. A committee of the U.S. National Academy of Sciences formed to study computer security issues began its report:

> We are at risk. Increasingly, America depends on computers. They control power delivery, communications, aviation, and financial services. They are used to store vital information, from medical records to business plans to criminal records.

> Although we trust them, they are vulnerable—to the effects of poor design and insufficient quality control, to accident, and perhaps most alarmingly, to deliberate attack. The modern thief can steal more with a computer than with a gun. Tomorrow's terrorist may be able to do more damage with a keyboard than with a bomb [National Research Council 1991, p. 7].

Scope of this Study

The National Research Council not only points out the pervasiveness of the risk, but they also indicate some of the problems. In the quote above, they mention poor design first. Poor analysis and design methods underlie such poor designs. We therefore focus on these analysis and design methods for secure computer-based information systems. This scope is broader than the design of a secure computing machine. For example, our concerns include those of electronic data processing (EDP) auditors who demand internal audits of application system designs prior to implementation [Weber 1988; Gallegos et al. 1987]. Our scope is broad enough to include the consideration of manual-processing elements as well as the security of computer-based information. We call this broad view the "information systems" viewpoint of the security issue and will show that it is critical to the development of safe organizational information resources:

> Because security is a weak-link phenomenon, a security program must be multidimensional... these involve physical elements and people as well as computers and software [National Research Council 1991, p. 65].

The importance of this information systems perspective of the entire application domain is also reflected in the widespread experience of information systems security practitioners. Some are troubled by the lack of such broad views in practice:

> What is missing in many instances is the involvement of concerned users, enlightened developers, experienced EDP auditors, experienced information security specialists, or other parties who understand how to incorporate controls into systems while the systems are still in development [Wood 1990, p. 13].

Our scope will be limited to matters of methodology in systems security development. Both *Webster's* and *Oxford* dictionaries define "method" as a procedure for attaining an object. This concept of method underlies much of the current research into IS development [Baskerville et al. 1992]. The limitation to matters of methodology does mean that the technical details of threats and system safeguards are excluded from this article. For example, some of the more familiar software safeguards, such as the organization of operation system security or the implementation of an encryption algorithm in the communications software, are not direct elements of our study. (This is analogous to excluding details of the organization of operating systems software from a study of a systems development method like dataflow diagramming.) Much of the work in modeling computer security (e.g., Landwehr [1981] and McLean [1990]) has been directed at this systems-software level. It is tangentially relevant only as a possible subcomponent to be included in the specifications of a secure information system. Likewise, work in Trusted Systems criteria (e.g., "The Orange Book" [U.S. Department of Defense 1985] and the IT-SEC European Standard [Commission of European Communities 1990]) is indirectly relevant only as possible specification criteria that may be imposed on specific hardware and software elements of an information system. Since our immediate concern is with the broader methods by which the specifications for secure information systems might be determined, we can usefully exclude the implementation details of the various security measures from the scope of our study.

Organization of the Concepts

We will require a taxonomy by which we can survey and compare many current security analysis and design methods. Because we will encounter strong indica-

tions that security concerns should be incorporated in all systems analysis and design methods, this framework must be meaningful to the general information systems audience. A simple taxonomy of methodological "generations" will relate the evolution of information systems security methods to the perspective of the broader information systems development community. By using the general characteristics of IS analysis and design methods, we can compare the evolution of security design methods to generations of broader information systems development methods.

The generation metaphor is useful because it allows a comparison of otherwise dissimilar methods by focusing on their intellectual evolution in response to a changing context. Certain conceptual aspects of the methods (such as assumptions or objectives) can thus be used for classification purposes. For example, we often speak of "generations of programming languages." Early generations were intent on machine efficiency; later generations focused on specific problem domain procedures, yet still later generations were intent on programmer productivity. The metaphor is particularly useful in comparing system development methods because a new generation evolves from the experience of its predecessor without necessarily implying obsolescence of the predecessor. Similarly, assembly languages (second generation), Cobol (third generation), and Prolog (fourth generation) still have appropriate applications as current programming languages. Nevertheless, they represent "generational" progress.

Unfortunately, we cannot draw upon a consensus about any scheme of generations, historic stages, or successive phases of systems development methods. The views are considerably varied. Nolan [1979] examines six stages of organizational systems development progress; Couger [1982] links five generations of systems development techniques to generations of computing hardware; Yourdon [1989] refers to three broad periods in the evolution of ideas and techniques in the field of systems analysis; Friedman [1989] describes three phases of computerization that are quite different from Yourdon's; and Hirschheim and Klein [1992] delineate seven generations. Other taxonomies deal less with the progression of systems development than with a comparative analysis of alternatives (see Avison and Fitzgerald [1988] or Davis [1982]).

This lack of consensus does not imply that these schemes are contradictory. Each operates within a level of abstraction that is appropriate for dealing with a particular problem set in the evolution of systems development. In this survey, we will be dealing with a different problem set (that of comparing systems development and systems security methods). For this purpose we will use a distinct three-generation scheme for comparing and contrasting security methods. We seek to describe parallel intellectual developments in the two arenas of IS development and security design, and we will describe concise features that embody the distinguishing characteristics of each generation.

Table 1 summarizes the comparison of three generations of systems development and security development methods. For each generation, this table notes the primary features, provides key examples of methods and tools, and references seminal security literature. Sections 1, 2, and 3 describe each generation noted in Table 1. The explanation of each generation has the following structure: First, we will establish links to the characteristics of overall systems methods that define each generation (Sections 1.1, 2.1, and 3.1). These characteristics include the primary objectives, means, challenges, and intellectual assumptions that concern each generation (Table 2). Second, we will employ these characteristics as a means of distinguishing those security methods that are members of each generation, providing a description and analysis of the key examples of published security methods (Sections 1.2, 1.3, 2.2, 2.3, and 3.2). Third, we will summarize the intellectual developments of each genera-

Table 1. Summary of Generations of Methods

Generations of Methods	Primary Features	System Development Methods and Typical Tools	Security Development Methods and Typical Tools	Seminal Security Works
First-Generation: Checklist Methods (1972 -)	Map of limited solutions onto the information problem	Vendor's technical sales procedures & literature	Security checklists & risk analysis	[Krauss, 1972] [Hoyt, 1973] [Courtney, 1977] [Browne, 1979]
Second-Generation: Mechanistic Engineering Methods (1981 -)	A partitioned complex solution that matches functional requirements	Top-down engineering, rapid prototyping, system and logic flowcharts	CRAMM, BDSS, control point and exposure analysis matrices, computer questionnaires	[Parker, 1981] [Fisher, 1984]
Third-Generation: Logical-Transformational Methods (1988 -)	Highly abstracted design expressing problem and solution space	Structured analysis, data modeling, information engineering, soft systems, data flow and entity relationship diagrams	Logical controls design, data flow diagrams	[Baskerville, 1988]

Table 2. Primary Objectives, Means, Challenges, and Intellectual Assumptions of Each Generation

Generation	Objective	Means	Challenge	Intellectual Assumption
First	Selecting components	Survey available elements	Mapping problem to solution	Universal solutions
Second	Partitioning the solution	Solve each functional requirement	Organizing and integrating a complex set of elements	Ideal, custom solutions
Third	Abstracting problem and solution	Model the essential attributes of the problem	Selecting the correct attributes for the model	Design in the abstract model

tion by analyzing the typical assumptions and the activities in the methodologies (Sections 1.4, 2.4, and 3.3). Fourth, we will consider the practical aspects of these methodologies by analyzing the typical strengths and weaknesses that characterize each generation of methods (Sections 1.5, 2.5, and 3.4). Because the third generation of security methods is underdeveloped, Section 3 concludes with a review of the important research directions raised by our analysis (Section 3.5).

Consideration of these research questions in an article aimed at a general audience is significant because those responsible for researching systems development methods seem to assume security is a separate issue: an implementation or computer science problem. We present an analysis that suggests

otherwise: systems methods will neither be trustworthy nor successful unless the general research regarding systems methodology incorporates security analysis design as an explicit objective. Hence, significant advances in specifying safe information systems will depend on better integration between two streams of research and practice: general IS development and specialized security methods. This article suggests a conceptual foundation for such integration in future research and practice.

Role of Benefits and Risk Analysis

Cost-benefit analysis is an important component in many general IS development methods. This component is of particular interest in first-generation methods, but declines in significance in the course of the development of second and third generations. There is a strong parallel to this component in systems security development known as risk assessment or risk analysis.[1] As a means of understanding these relationships, we will briefly describe the role of benefits analysis as a characteristic of each generation (concluding Sections 1.1, 2.1, and 3.1). The original development of risk analysis is detailed in Section 1.3 of the first generation of security methods. The important influences of this development on the computer-based second-generation methods are described in Section 2.3. The vestiges of this influence on the third-generation SSADM-CRAMM interface are described in Section 3.2.

[1]Authors from the auditing community (e.g., Gallegos et al [1987]) sometimes differentiate between risk analysis and risk assessment. To these authorities, *risk analysis* is the process of identifying risks and their characteristics, and *risk assessment* is the determination of the exposure to risk. When this is distilled to monetary units, this becomes *quantitative risk assessment*. This distinction is not critical for our purposes, and we can agree with the early risk analysis work (e.g., Courtney [1977]) from the computer security community that does not observe these delineations and assumes that the concept of *risk analysis* will include quantitative risk assessment.

1. FIRST GENERATION: CHECKLIST METHODS

The principal objective of the first generation of system design methods is the selection of the various solution components. The means of discovering the ideal system solution is to study the entire repertoire of available system elements. The challenge for all early system designers regards the mapping of a small population of known solutions onto the problem at hand. These methods arise from (and depend on) a highly limited set of elements that may universally be employed in the design project. It should not be surprising that early methods should share this intellectual characteristic, since the primal state of technology severely limited the solution space. Because the physical possibilities were so limited, these methods generally focused on concrete physical system specifications.

1.1 Checklist System Development Methods

The early checklist-style approaches to the definition of requirements and the specification of system design evolved from the experience of the early practitioners in the computer industry. They were often not methodical, using narrative language to describe systems and user requirements [Hawryszkiewycz 1988]. Both Nordbotten [1985] and Yourdon [1989] compare early attempts in system specification to literary composition, in Yourdon's words (p. 453): "Victorian novel specifications, hard to read, hard to understand, and virtually impossible to maintain." In those days, the only universal approaches to systems design were devised as technical sales procedures by computer manufacturers seeking to market their equipment widely. Ward and Mellor [1985] recognize that early methods were bounded because the technology was not available to solve any large set of typical problems. Manufacturers were forced to include applications and training with new computer systems [Friedman 1989]. The analysts and designers provided by the

manufacturers were inclined to start with the very limited set of possible solutions (computer systems available from their employer) and select the best inventory of system components suitable to the narrow information problem under examination. Essentially, the designers marked off the desired items from a checklist of possibilities. The practice broadened across the wider equipment marketplace when manufacturer-independent systems analysts appeared. But the checklist methods were similar, nevertheless: select the solution from the range of available computer hardware and software options that will ameliorate any aspect of the problem at hand. These early methods have been called "ad hoc" [Hawryszkiewycz 1988], "conventional" [Yourdon 1989], and "orthodox" systems analysis [Stamper 1979].

Checklist methods are still used in some arenas of information systems development. This method still prevails in the personal computer marketplace (where independent systems analysis is often not cost effective). Retail sales representatives often configure hardware and software solutions in small systems applications. These analysts map the available system elements (dictated by the wares stocked by the vendor) onto the customer's problems. The approach in large organizational end-user service centers, such as information centers, can be similar. In this case, the available solutions are called "supported" system configurations and are defined by allowed lists of approved suppliers, model numbers, and software versions.

The checklists do not reveal the benefits that an organization will receive from the selected set of design elements. As a result, the designer requires some economic means of differentiating those elements of the checklist that are needed from those that are not needed. That is, there should be some rational means for evaluating the affordability of the designer's choice of items from the checklist. Thus, checklist methods must encompass a cost-benefit analysis in which system element costs are compared with expected benefits. This analysis eliminates certain unprofitable items from the selected set of system elements that comprise the specification. This review stage becomes most important in the case of a vendor-provided or technologically self-serving analyst, where the largest possible set of solution elements holds the greatest direct benefit for the analyst. In such cases, many unprofitable components are likely to be included in the system without the controlling effect of a benefits analysis.

1.2 Checklist Security Development Methods

Checklist approaches to security systems design start with the solution selection task: the known controls that could be implemented. The analyst selects the best inventory of controls from the range of possible controls according to the information problem under examination. In both first-generation systems analysis and first-generation security analysis, the limited range of solutions (system elements or security controls) allows designs to be approached from the basis of "what can be done" rather than "what needs to be done." Evidence of this underlying philosophy is even found in early texts on computer security that offer a pedagogical organization based on a control taxonomy (e.g., Fitzgerald [1978]). Accordingly, checklist security methods generally do not begin with a view of what risks are involved in the case at hand—checklist methods begin their design with an examination of all known risks and controls. A list is provided of every conceivable control that can be implemented in a computer-based system. The analyst first checks to see if the control is already in place, analyzes its necessity when it is not found, and implements the control when required.

Many published security checklists were intended as guidelines for evaluating, not specifying, information systems security. Such an evaluation process, however, inevitably leads to the awareness of critical *lacunae* among system control elements. These checklists, like

Table 3. Organization of the SAFE, Computer Security Handbook, and AFIPS Security Checklists

SAFE Checklist Organization (844*)	Computer Security Handbook Checklist Organization (790)	AFIPS Checklist Organization (954)
1 Personnel Practices, (66)	1 Management, (10)	1 Planning and Risk Analysis, (45)
2. Physical Security, (161)	2 Risk Management, (25)	2. Physical Security, (184)
3 Operating Procedures, (76)	3. Employees, (24)	3 Backup and Recovery (85)
4 Backup and Contingency Planning, (63)	4 Legal Issues, (17)	4 Administrative Controls, (130)
5 Systems Development and Maintenance, (54)	5 Computer Crime, (82)	5 Systems Hardware and Software, (83)
6. Data Base Security, (101)	6 Contingency Planning, (20)	6 Communications (118)
7 Data Communications Security, (114)	7 Risks and Insurance (16)	7 Distributed Risk, (102)
8 Systems and Access Control Software, (89)	8 Auditing, (95)	8 Applications, (156)
9 Insurance, (17)	9 Application Controls, (101)	9 The Security Audit (51)
10 Security Planning and Administration, (45)	10 Hardware Security, (55)	
11 Application Controls, (58)	11 Facility Security, (32)	
	12 Monitoring and Control (53)	
	13 Software and Data, (16)	
	14 Data Files, (52)	
	15 Encryption, (7)	
	16 Communications Networks, (26)	
	17 Forms and Supplies, (82)	
	18 Contract Services, (50)	
	19 Mini and Microcomputers, (27)	

*The number of items in each checklist is noted after each list name.

the manufacturer's technical sales procedures mentioned above, thus became the core of ad hoc, nonmethodical approaches to the specification of information systems security.

Early texts on information systems security included sizable checklists in the appendices (e.g., Martin [1973], Hemphill and Hemphill [1973], Mair et al. [1978]), and there were an unknown number of checklists developed and attached to corporate information security policies (e.g., IBM [1972b]). At least three works are oriented entirely toward describing the security checklist. Table 3 summarizes the comparison of the organization, taxonomies, and sizes of these three important checklists. While these lists differ in their organization and exact topic titles, there is considerable consensus in many detailed items like communications encryption, disaster planning, off-site backup, and physical facility security. While space will not permit us to review these checklists in depth, the next few paragraphs note the characteristics and contributions of each.

SAFE Checklist [Krauss 1972; 1980]

This checklist is oriented toward an audit evaluation of an existing system. The SAFE checklist is characterized by its highly summarized and quantitative approach to system evaluation. The lack of explanatory material reflects an assumption that the designer is an experienced systems security analyst. The complex system of numeric scores and weighting factors allows a comparative analysis of systems (or perhaps comparisons of incremental stages in a system's security development). Patrick [1974] considers these numbers "intrusive" and suggests the checklist is useful as a design tool only if the analyst discards the numeric rating scheme.

Computer Security Handbook [Hoyt 1973; Hutt et al. 1988]

This checklist could be used to audit an existing system or proposed system design. It has substantial tutorial support. Each checklist is preceded by extensive

introductory material that discusses the principles and theoretical background of the checklist categories and entries. This background material reflects an assumption that the designer is an inexperienced systems security analyst with perhaps only one target system to manage.

AFIPS Checklist for Computer Center Self Audits [Browne 1979]

Like the SAFE checklist, this checklist is oriented toward the audit of an existing information system. In other ways, the AFIPS checklist strikes an interesting balance between Krauss and Hoyt. Krauss represented a highly quantitative detailed checklist with little introductory explanation. Hoyt is highly tutorial, and the checklists are briefer by comparison. Browne's work offers a brief introduction to the principles and rationale underlying each checklist area. His checklists are as extensive as Krauss's, but quantification is not encouraged. In his introduction, Browne briefly mentions a possible quantification scheme to use with his checklists, but he characterizes such schemes as "dangerous" oversimplification of the security decision. Perhaps the most significant advance in the AFIPS checklist lies in the manner of its original development. Rather than approaching the problem with a simple taxonomy of threats, this checklist was developed using a kernel-style model of threats and their usual defenses.

Hardware vendors also provided specific checklists. These lists had the advantage of linking both threats and controls to specific features of specific products. For example, IBM offered its customers an 88-point "Security Assessment Questionnaire" as a first step toward computer security. This checklist represented an improvement by providing cross-references to other IBM documents. The referenced documents then described the details of the controls [Fisher 1984]. Both the SAFE checklist and the AFIPS checklist are oriented to computer center audits.

1.3 Risk Analysis and Security Evaluation

Security checklists are ostensibly techniques for evaluating an information system's vulnerability. When these checklists become the core of an ad hoc first-generation approach to specifying security controls, however, then analysts need a formal cost-benefit model to help eliminate those control items that are rendered unprofitable by the specific situation of the particular organization under analysis. Risk analysis provides a formal, rational means for consistently evaluating highly specific vulnerabilities. It is an attractive means by which analysts can justify or reject various controls from an extensive checklist. Consequently, such quantitative exposure analysis techniques emerged early in the computer security community and in parallel with checklist evaluation tools (e.g., Martin [1973]). Risk analysis became very important in the context of first-generation security design because it provides a quantitative monetary value against which the analyst can offset the costs of proposed controls.

1.3.1 Elementary Information Security Risk Analysis

Courtney [1977] provides one of the most widely cited descriptions of the application of risk analysis in security controls justification. He defines two major elements of risk (R): P, the probability of an exposure occurring a given number of times per year, and C, the cost or loss attributed to such an exposure. Risk is calculated as

$$R = P \times C.$$

The probability P is determined with the aid of a Probability Range Table, which provides various subjective frequency times and an equivalent annualized "Loss Multiplier." For example, a subjective estimate of frequency of loss due to the risk exposure of an information system element might be "once in three years." This estimate yields an an-

nualized Loss Multiplier of 0.3333 (one-third of a loss per year). The cost/loss value C is estimated with the aid of a Cost/Loss Range Table, which provides the subjective cost of a single loss in decimal exponential steps, i.e., \$10, \$100, \$1000, \$10000, and so forth.[2] For example, an estimated loss of \$20,000 would yield a cost/loss value of \$100,000 (the next highest exponential increment of 10). The risk resulting from these examples of P and C above would be calculated to be \$33,333. This figure can then be used in assigning implementation priorities for new controls and cost, justifying any control changes in the system.

This technique is widely adopted or adapted by security methodologists, partly because the U.S. declared it to be a government standard [U.S. Department of Commerce 1979]. FitzGerald [1978] offers a revision of the technique which provides a finer granularity in estimating P and C.[3] Badenhorst and Eloff [1990] propose a security design method in which risk analysis is an integral part of the context for checklist design. A number of proprietary variants of such risk assessment methods have been developed that vary considerably in scope. These variants include those developed by Computer Resource Controls (Computer Science Corporation), Citibank [Saltmarsh and Browne 1983], and the National Computer Center in the U.K. [Wong 1977].

[2]The exponential scales for C and P permit the analysts to be "sufficiently inexact" to complete the task in a reasonable time Specific estimates can be refined later if a particular security course requires greater precision [Courtney 1977]. As the potential loss amount grows, this implies that analysts gradually lose their capacity to accurately estimate the loss. Any major loss probably would be accompanied by numerous incalculable side-effect losses. Similarly, low probabilities are forced exponentially into less and less precise ranges.

[3]Later work by FitzGerald recommended matrices and "Comparison Risk Ranking" as an alternative to monetary estimates [FitzGerald and FitzGerald 1990].

1.3.2 The Development of Risk Analysis Decision Support Systems

The quantitative nature of risk analysis makes an attractive target for development of computer-based decision support systems. Over the last two decades, more than 20 risk analysis packages have been introduced on the software marketplace, some of which did not survive [Ozier 1992]. Importantly, the research in this area has been a driving force in the development of automated second-generation security analysis and design methods. For the purposes of assessing the impact of this research, we can forego a detailed survey of all commercial packages in favor of just tracing the intellectual developments suggested by key examples from the scholarly literature.[4] For more software product details, the reader might wish to consult the *DataPro* or *Auerbach* product surveys or one of the government studies [Gilbert 1989; Communications Security Establishment 1990].

Securate

Securate is a computer installation security evaluation and analysis system [Hoffman et al. 1978] implemented in APL and tested at a number of commercial sites. Securate incorporates two important conceptual advances. First, it borrowed the security triplet concept from operating systems security and used this theory to model installation security. Securate triplets are composed of (1) a set of objects (installation resources to be protected, (2) a set of threats (intruder activities), and (3) a set of security features (protective measures). Importantly, this perspective means Securate is modeling security controls (the design), as well as threats. Second, this program dealt with the problem of imprecision in measurement of risk, value, and loss by using fuzzy sets of linguistic variables

[4]Several of these research articles detail work that ultimately evolved into a software product.

[Clements 1977]. Using Securate, analysts characterize security elements with linguistic variables from a preset vocabulary of values (such as "high" or "low") and modifiers (such as "somewhat" and "more or less"). Compatibility functions convert the linguistic values to a base scale (a probability curve). Each modifier is implemented as a function that receives the values and returns a geometric or linear modification of the values. Using this scheme, Securate calculates output ratings for security elements. These ratings are communicated to the analysts in terms of the linguistic values and modifiers of the input language. For example, the risk in the operating system subsection might be rated "More or less Medium."

Smith-Lim Approach

The Los Alamos Safeguards Systems Group [Smith and Lim 1984] constructed an automated assessment tool using Basic language programs. The notable conceptual characteristic was the use of more sophisticated data structures in risk modeling. This experiment used matrices (vectors) and trees. One matrix formed a model that mapped three generic "threats" onto four generic "targets." These two dimensions create nine[5] threat-target pairs (Figure 1). Using linguistic variables, each threat-target pair is evaluated using a hierarchy of event trees. Risk is assessed by plotting the evaluated vulnerability (absence of safeguards) and impact (severity of outcome) on another type of matrix, a linguistic algebra matrix map (Figure 1). The symmetry of this map can be adjusted to the preference of the organization. For example, an organization that prefers to be cautious about impacts might increase the upper values in the linguistic algebra map.

[5]The missing three pairs result from the assumption that environmental threats do not distinguish between specific targets.

Carroll-MacIver Knowledge Base System

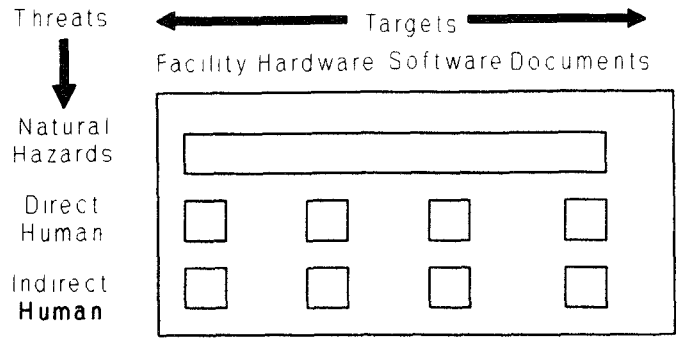
Carroll and MacIver [1984] exemplify advancements in risk analysis made by employing knowledge base technology for modeling computer security. They used Prolog to implement an experimental computer facility security certification system. This system is based on a two-dimensional model similar to Smith and Lim's. The Carroll-MacIver model contrasts "components" (personnel, hardware, processes, and information) against "attributes" (reliability, integrity, identification, access control, authorization, isolation, accountability, and detection).

Cost Effectiveness Model

Bui and Sivasankaran [1987] detail a straightforward Pascal implementation of a risk assessment decision support system. Simple risk analysis is used to rank the value of possible sets of control activities. An optimal control set is selected either by "minimized cost" or "maximized benefit-cost ratio." This program demonstrates an important conceptual characteristic by incorporating cost minimization as a major goal. Most previous systems had focused on simple risk measurement. Bui and Sivasankaran's work retains the basic purpose of risk analysis: achieving low risk at low cost.

LRAM

The Livermore Risk Analysis Methodology [Guarro 1987] exhibits two innovative conceptual characteristics. First, LRAM explicitly incorporates the Bayesian statistical effects arising from the potential failure of a security control. Second, LRAM employs a criterion hierarchy to isolate the threat assessment decisions from control selection decisions. The three criteria are MPL, LPI, and CBR values. MPL values are a simple integral scale of Maximum Potential Losses. Analysts use MPLs to assess the consequences of all threats. LPI values are Loss Potential Indicators derived by multiplying MPLs and control failure probabilities. Analysts use LPIs to identify all nontrivial threats that currently



Vulnerability	Impact				
	VH	H	M	L	VL
VH	VH	VH	H	L	L
H	VH	H	M	L	L
M	**H**	**H**	**M**	**L**	**VL**
L	H	M	M	L	VL
VL	M	M	L	VL	VL

Figure 1. Smith-Lim threat-target pairs and linguistic algebra map.

exceed the organization's risk threshold. CBR values are Cost-Benefit Ratios derived by multiplying LPIs and estimated threat frequencies. CBRs are used to prioritize new controls for implementation. The method is limited because it focuses on highly compartmented, individual control decisions. There is no mechanism for evaluating overall system security. The method also ignores the impact of a threat or a control across a costly aggregation of individually trivial losses.

Inversion Model Expert System

This knowledge-based system was constructed in both an expert system shell and Prolog [Baskerville 1988]. Its conceptual advance was in the use of a model founded on an historical analysis of trends in losses relevant to the automation of manual systems. The knowledge base suggests controls according to an *inversion model* (Figure 2) which evaluates the threats against data and process elements in manual systems that are evolving into automated systems. The inversion model finds that external risks (e.g., sabotage and espionage) are primarily threats to data in manual systems, yet primarily processing threats in automated systems. Conversely, internal threats (e.g., integrity and fraud) are primarily directed to data in automated systems, yet primarily threaten processing elements in manual systems. The knowledge base lists various resources, threats, and controls along with abstract security classes that comprise effective layers for control elements. The system will suggest minimum or maximum layers of controls as determined by the inversion model.

SPAN

This decision support system (DSS) provides suggestions to assist security plan

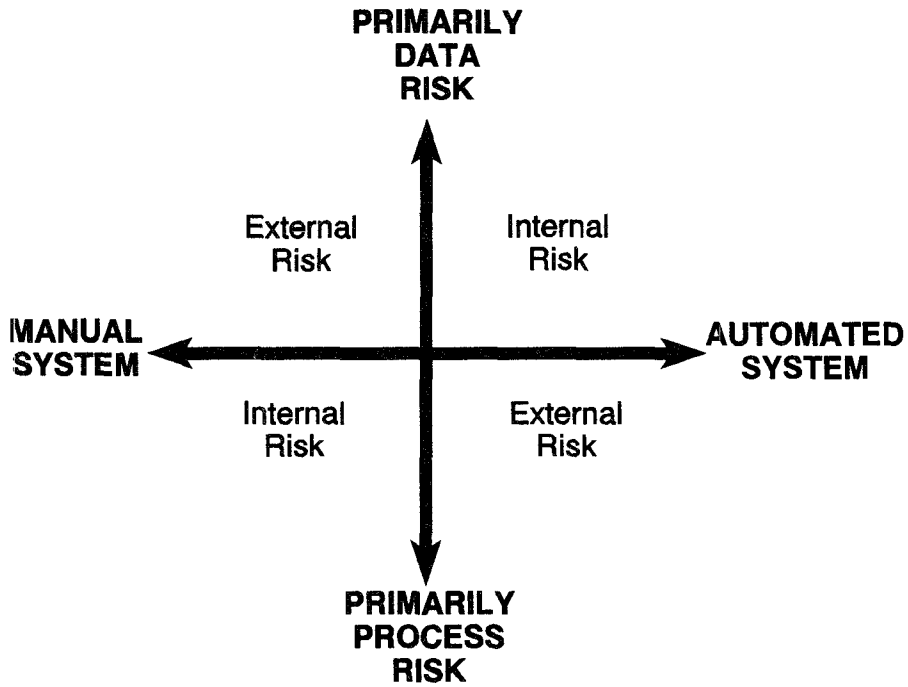


Figure 2. Inversion model of information security.

analysis (SPAN) [Zviran et al. 1990]. SPAN advances an approach that employs relational database theory as a conceptual foundation for modeling risk. The DSS includes a database module, dialogue module, and model module. The relational database details the interaction of resources, threats, risks, and countermeasures. Resources are associated with locations and classed as data, documentation, hardware, personnel, and software. Threats and countermeasures are independently associated with these resource classes and the distinct resource items. In addition, countermeasures are directly associated with locations, and the effects of these countermeasures are independently associated with threats. Risks are then associated with the location-threat pairs and the resource-threat pairs. Figure 3 diagrams these relationships.

1.3.3 The Challenges to Risk Analysis

While risk analysis originated as an essential feature of first-generation security design, it has been retained to varying degrees in later generations. Methodologists seem to have a penchant for risk analysis, but there are opposing viewpoints that challenge this technique as being basically impractical and unethical. These challenges are not recent [Glaseman et al. 1977; Nielsen and Ruder 1980], and they often consider the validity of probability operations on unverifiable single-point "guesses." Saari [1987] suggests that quantitative approaches to this design issue are presently unsuitable, in part because there are no reliable industry-wide statistics on which to base risk analysis. Indeed, Saari [1991] discredits most published statistics on computer crime.

Another practical problem arises because the basic data matrix is multidimensional (threats, assets, vulnerabilities, and costs). The population of data points implied by this matrix is immense. Computerizing the process may only magnify these problems with expansive improvements in the granularity of risk factors (e.g., Bayesian statistical models).

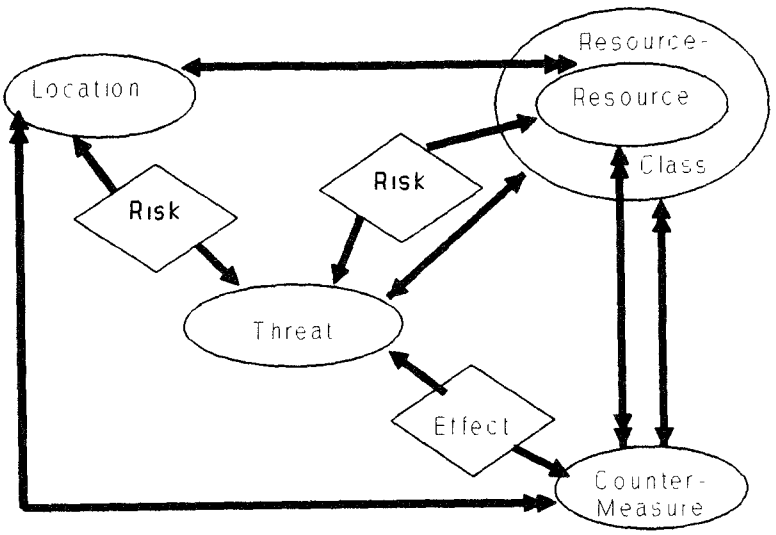


Figure 3. SPAN security DSS

These additional risk factors not only compound the matrix size, they may trigger philosophical paradoxes [Baskerville 1991]. For example, adjustments meant to improve the quantitative validity of risk analysis may destroy its interpretative validity without achieving the intended result. The "improved" method becomes invalid under both philosophies.

These practical problems are further confounded by the ethical conflicts that arise when certain security risks, such as privacy of personal data, are expressed in monetary units. Parker [1986] favors an alternative "Baselines Controls" approach that focuses on the concept of "due care" (rather than risk) in information management. Others, such as Guarro [1987] and Nielsen and Ruder [1980], convert risk problems to nonmonetary tokens to analyze risk comparatively.

These challenges do not go unanswered in the second generation, particularly in the automated methods. Some methods diminish the quantitative foundation of risk assessments by employing fuzzy variables. The BDSS technique, discussed in Section 2.3.3, uses more statistically valid measures (such as confidence factors and ranges) and an improved statistical reference database.

1.4 Assumptions and Activities of First-Generation Security Methods

We can identify several fundamental assumptions that characterize these first-generation techniques (Table 4). First, they assume the boundary of the solution space is most nearly defined by a highly limited set of useful solutions. Furthermore, they assume that it would be quite reasonable to construct an inventory of all possible controls that might be applied in the protection of information resources. Second, an even more fundamental presumption is that each member of that set of solutions will be universal to a large degree. This presumption surfaces in the expectation that a largely similar set of controls can be used effectively across a widely dissimilar population of problem systems. Finally, these methods assume the ultimate benefits of a system control can be expressed by a lowered probability of a threat occurrence or by the mitigation of the expenses caused by such an occurrence.

The implications of these first-generation assumptions entail three types of practical activities. The first type follows from the assumption of a limited inventory of universal controls. This leads de-

Table 4. General Characteristics of First-Generation Security Methods

First-Generation Checklist Methods			
Assumptions	**Activities**	**Strengths**	**Weaknesses**
▪ The boundary of the solution space is most nearly defined by a highly limited set of useful solutions. ▪ Each member of that set of solutions will be universal to a large degree. ▪ The ultimate benefits of a system control can be expressed by a lowered probability of a threat occurrence, or by the mitigation of the expenses caused by such an occurrence	▪ Survey checklist of security items ▪ Risk analysis ▪ Implementation	▪ Thorough examination of possible security components ▪ Lower expertise and training ▪ Low cost ▪ A wide variety of computer-based support tools	▪ Oversimplify more complex information systems ▪ Proclivity for unauthorized design shortcuts ▪ Documents are hard to understand and difficult to maintain ▪ Higher maintenance costs ▪ Overlook any innovative or new solutions ▪ Heavy dependence on risk quantification

signers to focus on a review of a universal checklist of security items. The purpose of such activities is to identify obvious, yet missing, controls. A second type is risk analysis. Risk analytic activities arise in the assumption that lowering threat probability or damage will represent the essential benefit of controls. The purpose of risk analysis is to identify a subset of these missing controls that can be justifiably proposed. The third kind of activity is implementation, to build everything that is affordable. The literature does not explore implementation carefully, implying that it involves straightforward, simple activities. This nonchalance is an implication of the assumption that well-known, universal controls will be applicable everywhere. Hence, construction details are not considered necessary.

1.5 Strengths and Weaknesses of First-Generation Security Methods

First-generation methods are marked by a conceptual foundation that is based on a predetermined set of possible system elements from which a subset is selected to solve a problem. A major practical strength of this method arises from the comprehensive survey of every conceivable element. Since checklists have been composed by recognized authorities in the field, this process will mandate a thorough examination of each possible system security component.

First-generation methods do not depend on deep analyst knowledge. The analyst or designer is guided, step by step, through a simplistic cookbook-style project. There is no physical or logical modeling, no diagramming, and no independent analysis. This characteristic reduces the level of expertise and training needed for an analyst to conduct a study. As a result, a strength in the first generation is the low cost of the security analysis and design projects.

These methods also entail a numerical basis in the checklist weights and risk analysis factors. This has led to a wide variety of computer-based tools, including expert systems, that assist with the checklist database and risk analysis calculations.

Among the major practical weaknesses, checklist methods oversimplify the security considerations that arise in more complex information systems. In

such systems, a single security consideration can pervasively touch upon many system components. The sheer volume of work required in a complex analysis may lead to unauthorized design shortcuts and consequent security oversights.

The unstructured, narrative-style security specifications are another weakness. These documents are hard to understand and difficult to maintain. Maintenance costs can be higher after a checklist-style security analysis. In addition, such cookbook approaches may overlook innovative, unique solutions to security control. Consequently, certain inexpensive yet effective controls may be given no consideration. Similarly, recent security technology may not be given due consideration since it may postdate the checklist.

Checklist methods are heavily dependent on risk analysis to help isolate unnecessary members of the solution subset. There are many critics of risk analysis, however (Section 1.3.3). In this light, heavy dependence on risk analysis also undermines checklist approaches.

2. SECOND GENERATION: MECHANISTIC ENGINEERING METHODS

The principal objective in the second generation of system design methods is the partitioning of complex system solutions. These methods assume the most effective means for discovering the ideal system solution is to identify and solve each detailed functional requirement. Discrete solution elements then can be merged into a coherent system. Technology no longer limits the range of solution elements. A wide range of computers, peripherals, and software components can be used to construct solutions.

The challenge for system designers in the second generation is to detail the organization of a complex array of diverse elements that are to be integrated into a single-solution system. Rather than assembling the best-fit system from the available components, second-generation methods set about building an ideal custom solution. This focus on complex customization is the primary intellectual assumption that distinguishes the second generation. Engineering concepts dominate the methods, and designers follow mechanistic procedures to organize and partition the requirements into a collection of physical components (such as devices, data media, or work flow). While analysis and design activity generally focuses on this physical specification, note that the organizing and partitioning principles do represent a shift in focus. This shift is a movement from a total concern for the physical elements (checklists) toward certain overall conceptual ground rules for design. Table 2 compares the primary objectives, means, challenges, and intellectual assumptions of the second-generation methods to other generations.

2.1 Mechanistic Engineering System Development Methods

In the second generation of systems development methods, designers expect that a detailed examination of functional requirements will lead to the specification details for a new system. The methods focus on the production of mechanical specification of input, storage, and output formats, along with details of procedures needed to transform input or storage into outputs (e.g., Waters [1973], Shelly and Cashman [1975], and Murdick [1980]). Mechanistic engineering methods are usually framed by the classical project life cycle, also called the "waterfall" or "bottom-up" approach [Yourdon 1989]. Common tools include system flowcharts, record layouts, print charts, and program logic flowcharts. These methods are chiefly concerned with how the proposed solution will be constructed. Ward and Mellor [1985] characterize these methods as "implementation dominated." Lucas [1976] also notes that these methods are oriented to transaction-processing systems and paperwork automation.

These methods are found in the mainstream of information systems development projects today. Not only are they

found in current waterfall projects, but also in newer requirements definition techniques. Most forms of rapid prototyping, for example, depend primarily on a working, concrete model of the proposed system as a design artifact [Agresti 1986]. When there is no higher-level abstract model (characteristic of the third generation), prototyping will force the designers to work entirely in the context of the physical system specifications. For example, requirements are often partitioned into the forms, reports, and menus that database software provides. Such simple instances of prototyping are characteristically second generation.

Like checklist methods, mechanistic engineering methods also have a weak link between the solution design and the benefits to the organization. This gap arises because the analysis of the problem is isolated in a separate preliminary stage. Most analysis and design continues to focus on the technical aspects of system solutions. That is, there is a poor linkage between the requirements analysis and the specification of design. Thus, the potential mismatch between organizational needs and the information systems solution continues as a worrisome, though somewhat reduced, possibility in second-generation systems methods. Cost-benefit analysis is usually a holdover to assure that the technical solution remains linked to organizational needs. Benefits analysis prevents a lengthy system development project from pursuing a vague problem definition. It is a reminder to designers that they must keep the solution in balance with the problem.

2.2 Mechanistic Engineering Security Development Methods

Mechanistic engineering methods also inhabit the IS security development world. These second-generation methods are clearly distinguishable from checklist methods by their initial focus on system requirements. The engineering perspective means that security specifications will not spring directly from any index of available technologies.[6] In second-generation security analysis methods, the techniques focus on physical specifications like control points and access procedures, along with details of existing or potential control processes that protect system elements. A distinct mechanistic engineering life cycle exists—a security waterfall that begins with asset and threat analysis and moves toward security maintenance (Table 5). These methods also have the same clear orientation to the transaction-processing systems found in the more general second-generation systems analysis methods.

Second-generation security methods inherit risk analysis and the inventory checklist of possible controls from the first generation. Risk analysis remains as a cost-benefit holdover from the first generation, just as in the more general systems methods. In addition, second-generation methods suggest that the first-generation checklists are a good source for ideas as to what kinds of controls can be specified in system security design. The computer-supported second-generation methods described below provide exposure analysis calculations and controls databases that essentially automate these inherited first-generation steps.

2.2.1 *Fisher's Approach*

Fisher's [1984] *Information Systems Security* is one of the first comprehensive requirements-oriented methods for the design of data security. He builds heavily on IBM experience, expanding Courtney's risk analysis into a complete, mechanistic engineering approach to IS security. Notably, Fisher avoids the controls

[6] These methods sometimes reference the earlier checklists as sources for suggested control technologies once the exposure points are clearly known. Nevertheless, this controls inventory is no longer the fundamental means of organizing the security design effort. Checklists are only useful after the critical exposures are identified, and then only for locating effective controls for the given function.

Table 5. Generic Second-Generation Security Project Stages

Stage 1	Identify and evaluate system assets
Stage 2:	Identify and evaluate threats
Stage 3	Identify possible controls
Stage 4.	Risk analysis
Stage 5:	Prioritize controls for implementation
Stage 6·	Implement and maintain controls

checklist details that dominated the previous literature on the subject.

Fisher defines a waterfall-style five-phase design method that follows the establishment of a security/asset protection organizational policy. The security administrator implements a data security plan that includes a vital-records plan, an access control plan, an emergency response plan, an interim processing plan, a restoration plan, and a data security classification program. The five phases are:

(1) *Define the Data Inventory.* This phase is likely to be unnecessary in well-designed information systems. Database definition is considered to be a "remedial" step when required.

(2) *Identification of Exposures.* Fisher structures this process by mapping Courtney's [1977] six exposure groups onto the 11 data exposure control points suggested by IBM [1972a]. (Table 6 illustrates these exposure groups and control points.)

(3) *Assess Risk.* This step is a variant of the Courtney risk analysis process described in Section 1.3.1.

(4) *Design Controls.* Fisher uses Courtney's [1977] characterizations of controls as "preventative," "detective," and "corrective." For each risk, an appropriate class of control should be selected. A particularly damaging risk, for example, would call for a preventative control. A minor risk may require only detection because the damage is easily repaired.

(5) *Analyze Cost Effectiveness.* This phase is a capital investment view of risk analysis. The risk figure is treated as the "Return on Investment," and the control figure is treated as the "Investment." Management makes decisions regarding which controls are to be implemented based on both a maximum acceptable level of risk and a minimum acceptable return on investment.

Fisher's work embodies a conceptual advance in its use of the control point exposure grid as a primary analysis vehicle. The implementation details of the existing system, not a controls checklist, are now the primary vehicle for security analysis. In addition, Fisher discusses a more complete security life cycle, called Span, in an appendix on security-planning methods. This planning process incorporates all phases of a typical systems life cycle: (1) determination of safeguards (analysis and design), (2) implementation of safeguards (implementation), and (3) monitoring of controls and operations (operations and maintenance).

2.2.2 Parker's Computer Security Program

Parker [1981] developed this method for the Computer Security Institute and tested it at SRI International. Like Fisher, Parker stresses the need for a proper management environment for information systems security. Parker suggests, however, that either a task force or permanent organizational function could decide on the scope and conduct of the security review. His method comprises five phases:

(1) *Identification and valuation of assets* is facilitated by a model that helps pinpoint assets based on type of asset (e.g., people, supplies, hardware, data), form of asset (e.g., moveable property, magnetic patterns, printed paper), location of asset (e.g., remote, internal environment, computer), and

Table 6. Fisher's Exposure-Identification Structure

Exposure Groups	
accidental disclosure	intentional disclosure
accidental modification	intentional modification
accidental destruction	intentional destruction

Data Exposure Control Points

(1) *Data gathering* - The manual creation and transportation of data, i.e., the point of inception of data into the information system.
(2) *Data input movement* - The manual movement of source documents to the work area.
(3) *Data conversion* - The source documents are converted to machine-readable form.
(4) *Data communication (input)* - The transmission of the machine-readable data.
(5) *Data receipt* - The receipt and storage of data, either by electronic communications or manual facilities.
(6) *Data processing* - The execution of application programs to perform the intended computations.
(7) *Data preparation (output)* - The preparation of data output media such as tapes, disks, diskettes or microforms.
(8) *Data output movement* - The manual movement of computer-produced output.
(9) *Data communication (output)* - The actual transmission (electronic or manual) and delivery of output to the user.
(10) *Data usage* - The use of the data by the user, including storage of the data while in use.
(11) *Data disposition* - This point covers the disposition of data after the usage period.

accountability (e.g., EDP department, vendors).

(2) *Identification of threats* is aided by the application of a threat model to ensure a complete and orderly inventory. This model identifies threats by source (e.g., data processing employees, employees, vendors, outsiders, natural forces), motives (e.g., incompetence, human failure, irrational behavior, crime, advocacy), act (e.g., covert, multiple events, physical, logical, local/remote, collusion), results (e.g., disclosure, destruction), and losses (e.g., monetary, denial of use, personal values, health/life, privacy).

(3) *Risk assessment* can be performed using the Courtney risk analysis method described in Section 1.3.1. Nevertheless, Parker suggests that exposure analysis is a less costly alternative that eliminates both guesswork and consensus determination. Exposure analysis bases its loss projections on the numbers of people who could accidentally or intentionally cause losses. This analysis plots occupations with such access capabilities in a matrix (Table 7) against the potential damage to vulnerable assets (e.g., disclosure, denial of use). After monetary values are assigned to each vulnerable asset, the risk exposure of all assets can be computed by occupation. An ordered list is then created which details the risk level of each occupation based on the range of assets exposed to that occupation. Management then applies the criterion of a "prudent person" in deciding the degree of implementation for the exposure list.

(4) *Safeguards identification, selection, and implementation* involves safeguard selection principles such as cost effectiveness, minimal reliance on human intervention, fail-safe defaults, override safeguards, and independence of control and subject. By "independence," Parker means keeping those concerned with safeguards remote from those constrained by safeguards.

(5) *Implement and advance the program.*

Table 7. Parker's [1981] Exposure Analysis Matrix

Occupational Vulnerability Analysis

	VULNERABLE ASSETS														
	--Data--			Application --Programs-			Systems --Programs-			Computer --Equipmt-			System Service		
OCCUPATIONS	M	DE	DI	M	DE	DI	M	DE	DI	M	DE	T	T	DN	
Tape Librarian		4	4		3	3		3	3		1	1			
Data Entry Clerk	2	2	2								1	1			
Operator	1	5	5		5	5		5	5		5	5	5	5	
Operations Manager	1	5	5		5	5		5	5		1	1		5	
DB Administrator	3	3	3								1	1			
Systems Programmer		5	5		5	5	5	5	5		1	1	5	5	
Applic. Programmer	1	1	1	2	2	2					1	1			
EDP Auditor	5	5	5	5	5	5	5	5	5	5	5	5	5	5	
Systems Engineer							5	5	5	5	5	5	5	5	
Security Officer		5	5		5	5		5	5	5	5	5	5	5	
EDP Auditor	5	5	5	5	5	5	5	5	5	5	5	5	5	5	

Acts Codes	Exposure Scale
M - Modification	blank - no effect
DE - Destruction	1 - up to 20%
DI - Disclosure	2 - up to 40%
T - Taking	3 - up to 60%
DN - Denial of Use	4 - up to 80%
	5 - up to 100%

Parker suggests that newly selected safeguards can be organized for implementation by both cost and urgency. The valuable information collected during the review should be maintained and updated as an ongoing process of system adaptation.

Parker makes several important contributions with this method. First, the bottom-up life cycle is clearly essential to the method. (Fisher treated the complete project cycle only briefly in an appendix.) Second, Parker recognizes the importance of social aspects of security and balances these aspects against the technical problems. Working from his earlier studies in computer crime, Parker [1976] concentrates on "motives," "acts," and people as "sources" for the identification of threats. The exposure-analysis matrix orients the analysts to the occupations of "people with access." In this regard, Parker's techniques fundamentally conflict with the study by Hoffer and Straub [1989] that found no significant relationship between user privilege and system abuse. This may mean that analysts will be distracted from controls that protect against outsiders or natural threats. Finally, Parker's program is among the first to incorporate a qualitative alternative to risk analysis (the prudent-person criterion). He admits, however, that traditional risk analysis may be required within his method.

2.3 Computer-Based Mechanistic Engineering Methods

Most of the fully computer-supported security analysis and design methods evolved directly from automated tools for risk analysis. These methods have gone beyond computer-based modeling of the risk decision. They support typical life cycle activities such as data gathering, alternatives generation, and maintenance. These software packages also provide an extensible database of possible threats, assets, and controls. The analyst selects a subset of these elements during the analysis and design process. As a corequisite of automating the mechanistic engineering method, these packages have necessarily automated both the risk

analysis and the checklist foundations that mechanistic engineering methods inherited from the first generation.

2.3.1 CRAMM

"CCTA's Risk Analysis and Management Methodology" is important because the U.K. Government Central Computer and Telecommunications Agency (CCTA) adopted this method as a government-wide standard approach to risk analysis and security management. This adoption could lead to international acceptance. The method comprises three stages, each of which is supported by the CRAMM software [Farquar 1991].

Stage 1

During this stage, analysts set the scope and the boundaries of the study. The exact set of physical assets (and associated data or software assets) is delineated. The analysts then conduct structured interviews with each data "owner." When the owners are identified, they are asked to qualitatively evaluate each of their assets. They not only contribute estimates of the asset values, they also estimate the effect of problems on a 10-point scale. For example, they use a software-generated scale to enumerate the impact associated with interruption or destruction of each physical asset and the modification or disclosure of each data asset.

Stage 2

The second stage commences with the grouping of assets into suitable "asset groups." Asset groups become the basis of further analysis. Using a database of 17 generic threats, the software selectively generates up to 32 threat and vulnerability questionnaires for each asset group. The owners then evaluate the vulnerability of each asset to a particular threat on a high-to-low scale. The CRAMM software uses this data to calculate each asset group's level of risk on a 5-point scale. Before continuing to stage 3, the team undertakes a rationalization step in which the risk levels are reviewed qualitatively. If owners misinterpret questions, their responses may lead to irrational risk levels. The risk levels must be holistically reviewed and assessed for this problem.

Stage 3

In the third stage, the existing system countermeasures (controls) are entered into the CRAMM software. Based on asset groups, risk levels, existing controls, and an internal database of 900 possible countermeasures, it then compiles a list of recommended additional countermeasures.

Once the CRAMM process is complete and the recommended countermeasures have been implemented, the organization enters a CRAMM database maintenance cycle. This cycle can include both routine and event-driven reviews. Each review is a smaller-scale iteration of the entire CRAMM process. The reviews are used to update the database. After updates, the analysts must recompile the countermeasure recommendations. Changes in the recommended countermeasures are then implemented.

Clearly, CRAMM contributes a computer-supported mechanistic engineering method for security analysis and design. It follows all of the generic stages in second-generation security methods (see Table 5). It is also distinguished by the high level of expertise required to use the package. CRAMM assumes a well-trained and experienced system security analyst will conduct the study. For example, for the data-gathering activity, it assists the analyst in generating paper-based questionnaires. These paper questionnaires are then completed and returned by the users. A trained analyst must manage and review every step of this process. The bulky CRAMM reports are highly technical and tend toward the narrow security jargon that requires interpretation by a professional.

2.3.2 RISKPAC

Two major objectives of RISKPAC are ease of use and ease of results interpre-

Table 8. RISKPAC Questionnaires [Computer Security Consultants 1988]

Questionnaire	Unit of Analysis	Scope of Enquiry
Application	end user	impact on business functions
Physical Security	location	physical plant weaknesses
Personal Computers	building	environment and applications
Telecommunications	network	communications risk
Audit Risk	computer environment	need for auditing
Computer Systems	each computer	operating system environment
Computer Application	each application	operational risk and sensitivity
Trusted Computer System Evaluation	each system	defense standards compliance
Product	the organization	products and services
Computer requirements	each application	loss value and recovery requirements
Location	each location	disaster exposure
Recovery Options	operational function	specific requirements for recovery

tation. Thus, the method can be used by system security analysts with only moderate levels of training and experience. This "risk profile analyzer" program provides various interactive questionnaire sessions. Security designers, system professionals, and information system users can all interact directly with the program. The questionnaire scheme orients toward qualitative user evaluations by employing linguistic variables, and the final analysis is presented in qualitative terms. The RISKPAC process requires a survey, a profile table, mapping files, and requirements [Computer Security Consultants 1988].

Survey

The RISKPAC survey gathers information about the environment, organization, system, and applications. It uses the 12 questionnaires shown in Table 8. This table also lists the basic top-level unit of evaluation for each questionnaire (end user, location, etc.). A questionnaire is completed for each unit of analysis or evaluation. Each questionnaire may have one or two additional levels of evaluation. For example, level one of a questionnaire may ask about a computer room; level two may ask about a computer in that computer room; and level three may ask about an application on that computer. Table 8 also gives a brief summary of the topical scope of each questionnaire. In addition, there are five highly specific questionnaires for use in banking, manufacturing, insurance, federal government, and state government organizations.

Profile Table

Entries in the profile table are triggered by the questionnaire scores. This table defines risk categories and relevant exposures. Each entry in the table includes weights and measures that rank risk from low to high (Figure 4).

Mapping Files

This component embodies a map between the profile table entries and the standards file. The standards file contains the database of controls.

Requirements

This element is the central output of RISKPAC. It includes explicit prescriptions for controls, beginning with a sim-

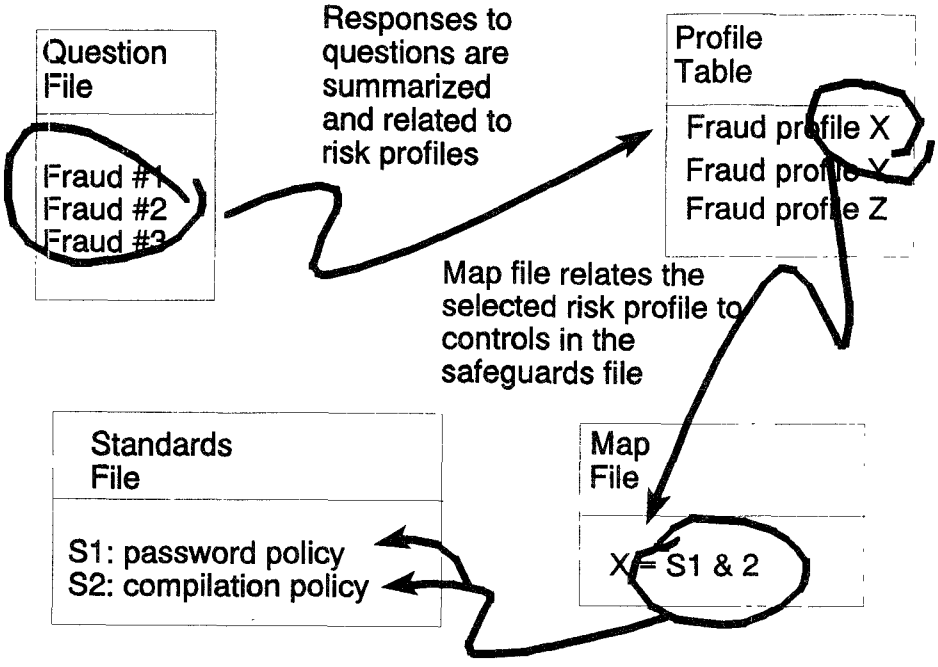


Figure 4. Components of the RISKPAC method [Computer Security Consultants 1988].

ple baseline component representing a minimum security recommendation for the organization. Further security recommendations are based on the organization's unique risk profiles and utility functions (as derived from the questionnaires).

The RISKPAC method seeks to balance quantitative and qualitative analysis. In the RISKPAC authors' view, its evaluation orientation is toward the qualitative realm of fuzzy sets and linguistic variables. Nevertheless, the package now incorporates an earlier companion product that supports quantitative risk calculation of an annualized loss expectancy adhering closely to Courtney's [1977] original proposal. This feature is intended to permit U.S. government agencies to use the program and still adhere to the federal standards [U.S. Department of Commerce 1979].

RISKPAC is distinguished by several fundamental advances. First, it seeks a primary foundation in utility theory rather than the more common Bayesian statistical theory. Second, it openly seeks to use more practical qualitative terms in calculating the security requirements. Finally, in relationship to other second-generation methods, RISKPAC requires less high-cost professional labor in the security analysis and design process. This reduction in professional labor is accomplished not only by the use of qualitative terminology but also by interacting directly with the users and other system professionals in the data-gathering activities. These advances mean RISKPAC might be more appropriate for smaller organizations that lack the security expertise required by other methods.

RISKPAC also exhibits several distinctive drawbacks. It is difficult to control the analysis and design process because it is so heavily software dependent. Virtually all major second-generation activities are governed by the mysteries of RISKPAC computer algorithms. A RISKPAC manager program enables organizations to tailor the questionnaires, and there are trace indicators on the recommended controls. The exact utility

functions are secreted in profile tables and map files. This characteristic means that there is a limited capacity to model changes in organizational systems security. RISKPAC will calculate a descriptive net present value of any change, but it does not support comparisons between various security strategies. Another drawback may lie in unrealistic expectations about conducting a security review with inexperienced analysts, since analyst experience may be critical in making valid security design decisions [Baskerville 1991].

2.3.3. BDSS

The Bayesian Decision Support System is a complete computer-supported information security design method that has evolved directly from quantitative risk analysis techniques [Ozier 1989]. While other second-generation methods have diminished risk analysis with softer qualitative evaluations, BDSS is remarkable because its developers have progressively increased the formal and statistical rigor of their method. The BDSS security design process is organized into nine software-supported activities. These activities group into four iterative stages or phases.

Data Gathering

This stage comprises three activities. First, a *project-sizing* module uses a text editor to elicit project language for scope, objectives, risk acceptance criteria, etc. Second, an *asset/inventory/loss valuation* module uses input screens to inventory and evaluate the information system elements. Third, a *threat-vulnerability-mapping* module uses a series of up to 94 interactive questionnaires to capture qualitative and quantitative data regarding each threat exposure.

Machine Risk Analysis

This stage comprises two computing machine activities that may be invoked iteratively as a background process by other activities. First, an *impact analyzer* compiles a quantitative risk model that includes frequency and exposure distributions for threat-asset categories. This analysis may involve locally developed threat frequency distributions or the internal BDSS database of threat frequency distributions.[7] Next, a *risk analyzer* applies a rich set of statistical algorithms to this model and develops both risk curves (Figure 5) and annualized exposures for each threat.

Interactive Safeguards Analysis

This process involves three iterative computer-based activities. First, a *safeguard analyzer* displays the threats that were discovered during threat vulnerability mapping. The package suggests safeguards from its database of over 1900 safeguards for possible selection by the system security analyst. Second, a *safeguard cost/benefit* module collects information regarding the costs of safeguards for further use in identifying the most economically attractive set of controls for implementation. Finally, an *evaluate-and-revise* module compiles the frequency distributions, the exposure data, the risk curves, and the effects of selected safeguards. This evaluation enables the designer to review the organization's overall threat profile, as well as discrete threats and safeguards. The software will rank the data to further aid in the safeguards selection process.

Report Generation

BDSS provides a report generator that produces three major types of documentation, each type including relevant graphs. An *executive summary* provides a high-level report on the design process, focusing on significant vulnerabilities. The *decision support* documentation contains the relevant information that leads to a defensible decision to implement or forego each security control. The *technical analysis* provides the complete set of

[7] Threat frequency distributions are for North and Central America and Hawaii.

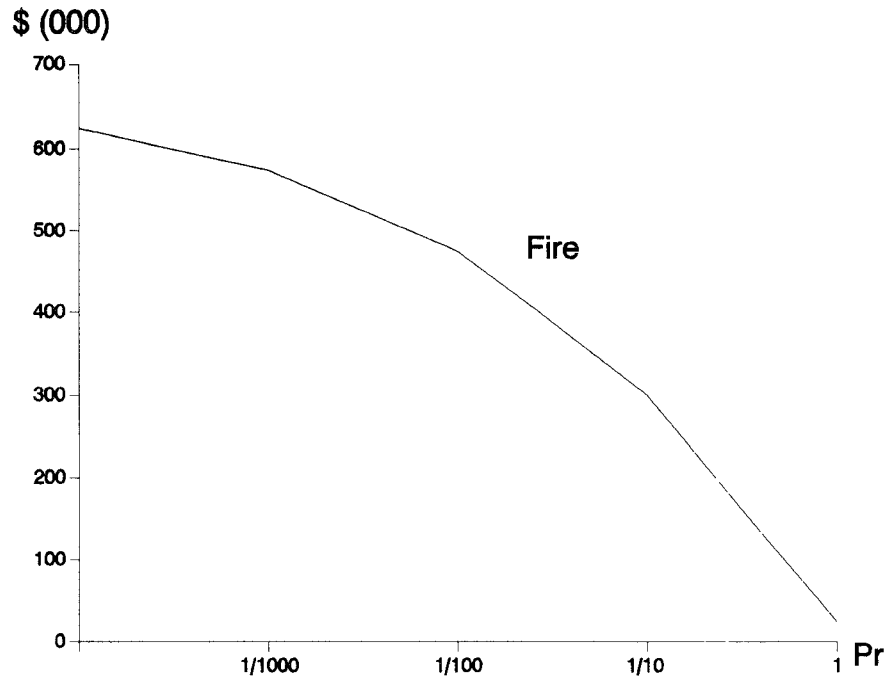


Figure 5. A risk curve or risk profile graph.

detailed documentation that the security analysis and design project developed.

BDSS makes an important contribution in the manner by which it holds tenaciously to its quantitative rigor. The original risk analysis approach and many of its recent variants use highly judgmental and exponential single-point values for threat probabilities and costs. BDSS uses nonexponential ranges for these values. The original risk analysis approaches do not distinguish between catastrophic and mundane threats.[8] BDSS attacks this problem by making such comparisons with risk-profile graphs (Figure 5). The original risk analysis approaches depend on expert opinion for threat-likelihood estimates. BDSS uses a database of over 60 threats determined with an 80 percent confidence factor by a team of experts using data from sources such as FBI crime statistics, the U.S. National Fire Protection Association, etc. These features place BDSS squarely in opposition to the critics of quantitative risk analysis (discussed in Section 1.3.3). It has continued to advance information security design in concert with advances in the field of quantitative operations research.

2.4 Assumptions and Activities of Second-Generation Security Methods

Four fundamental assumptions characterize second-generation techniques (Table 9). First, the requirements and impacts of the security system elements will be complex and interconnected. There is a web of assets, threats, and safeguards that must be partitioned for comprehension. Second, the exact controls could be unique and ideal for the system requirements. This uniqueness requirement is evidenced by the need to make the con-

[8] In the original risk analysis approach, a catastrophic event that destroys a billion-dollar corporation occurring every ten thousand years has the same quantitative risk exposure as a more mundane risk that might destroy a $500,000 asset occurring every five years.

Table 9. General Characteristics of Second-Generation Security Methods

Second-Generation Mechanistic Engineering Methods			
Assumptions	**Activities**	**Strengths**	**Weaknesses**
▪ The requirements and impacts of the security system elements will be complex and interconnected ▪ The exact controls could be unique and ideal ▪ The feasible solution set is not bounded ▪ A well-understood and well-documented security design leads to efficient security maintenance and modification	▪ Inventory assets & threats ▪ Enumerate possible controls ▪ Risk analysis ▪ Prioritize controls ▪ Implement & routine review	▪ Comprehensiveness of the approach ▪ Detailed and well-organized documentation ▪ Reduced operation and maintenance costs ▪ Useful for very complex systems ▪ Impact of modifications easily assessed	▪ Complex design process ▪ High degree of training required ▪ A design team needed ▪ Higher costs ▪ Preexisting system or specification ▪ The functional and security design are isolated ▪ Vestiges of risk analysis ▪ Irrational exposure estimates

trols and threats databases extensible, allowing for unique additions during the security analysis and design process. This leads to a third assumption, that the solution set is unlimited and impossible to capture in a single database or checklist. Possible target systems and technologies are complex and varied, and they cannot be adequately protected by any generic set of controls. Fourth, second-generation security methods inherit from the more general systems methods the assumption that a well-understood and well-documented security design will lead inevitably to more efficient security maintenance and modification.

In the second generation, these assumptions lead to five general types of activities. First, analysts concentrate on detailed inventories of system assets and potential threats. This activity uses various analytical techniques to reduce the complexity implied by the assumption that security elements will be complex and interconnected. Second, analysts must enumerate all possible controls. Such activities follow from the assumption that designers will have to select a unique and ideal control set for the system. Third, analysts conduct risk analysis. Fourth, they prioritize controls. These two types of activities are implications of the assumption that there are no boundaries on the set of possible controls. Designers can specify safeguards endlessly. Consequently, the risk arithmetic is used to prioritize controls that can be constructed according to an implementation plan. Finally, the assumption that security maintenance is critical leads to the fifth type: maintenance review. This kind of activity entails more than just controls implementation. It also includes an ongoing maintenance cycle or routine security review.

Table 10 compares these five types of activities to counterparts in the classical life cycle. Note the heavy emphasis on security analysis activities and the light regard given to controls design activities. This emphasis is consistent with the assumption that the controls solution likely will be composed of generic universal controls. The design is essentially detailed in the analysis documentation of newly cost-justified controls.

2.5 Strengths and Weaknesses of Second-Generation Methods

Mechanistic engineering methods focus on the technical functionality of system requirements. In security methods, this focus emphasizes the identification of

Table 10. Comparison of Classical Systems and Security Activities

Classical Life Cycle Activities	Security Life Cycle Activities
Investigation & Data Gathering	Inventory Assets & Threats
Alternatives Formulation	Enumerate Possible Controls
Cost-Benefit Analysis	Risk Analysis
Select Alternative	Prioritize Controls
Implement & Maintain	Implement & Routine Review

critical exposures at physical points in the system. Mechanistic engineering methods adhere to the theory that detailed system security requirements provide the ideal foundation upon which to launch a security design. A practical strength of this idea is the comprehensiveness of the approach. These methods will suit highly unusual systems and more common types of systems because the analyst is led to develop solutions unique to the functions of the system at hand. To some extent, the methods also gain strength by not surveying an exhaustive list of controls, many of which will not be relevant to every type of system and need only be rejected.

The detailed and well-organized documentation is also advantageous and can reduce costs for operating and maintaining the system controls. These methods are also suitable for complex, expensive computer-based information systems, since they link controls directly to each individual processing element within the system. Analysts can easily identify and compensate for the security impact of any considered modifications to the information system.

Major weaknesses arise from the complexity of the design process. Among these weaknesses are the high level of training required of the systems security designer. They must be "super analysts" who understand the entire functional specification and can contrive controls that will protect each processing element. Pragmatically, these methods recognize that it is more likely a design team will be needed to effectively conduct a security study. This team requirement means higher labor costs, which when added to the overall complexity and uniqueness of the design, may mean considerably higher overall costs of the initial security analysis and design.

A different kind of weakness stems from the predication of mechanistic engineering security design methods upon the preexistence of a complete operational system (or its specification). This predication means that the functional design and the security design are conducted with a high degree of isolation. Ultimately, conflicts can arise if the security designer fixes "vulnerabilities" that the functional designer purposely opened as "features." Such conflicts may destroy the system [Baskerville 1992].

Finally, these methods retain the vestiges of risk analysis and may further compound some of the risk analysis weaknesses discussed in Section 1.3.3. The exposure analysis in these methods is weakly grounded. There is no empirical validation of the linkage between occupational privilege and computer abuse [Hoffer and Straub 1989]. Especially note that experience with computer-based mechanistic engineering methods indicates that involving users in the estimation process leads to irrational estimates of risk exposure [Farquar 1991].

3. THIRD GENERATION: LOGICAL TRANSFORMATIONAL METHODS

The principal objective of third-generation system design methods is the abstraction of the problem and solution space. The most effective means of discovering the ideal system solution is achieved by modeling the essential attributes of the information problem and

its solution. The most important challenge to designers in the third generation regards the correct selection of attributes that should be abstracted in the model. A major intellectual development in these third-generation methods is embodied by the presence of high-level abstract design stages. During one or more design steps, the system is removed from its dependence on concrete technology. Certain design problems are resolved in a highly abstracted model. This model permits the designer to express attributes of both the problem and the solution is an abstract design space that demonstrates an intended match. Third-generation methods can be distinguished from first- and second-generation methods because, in the earlier two generations, the analysis and design tasks are entirely rooted in physical system elements. The shift in focus noted in the second generation has continued, leaving behind physical concerns for an intense focus on conceptual and logical design. Table 2 compares the primary objectives, means, challenges, and intellectual assumptions of the third-generation methods with the other generations.

3.1 Logical Transformational Systems Methods

Conventional second-generation methods may assume too narrow a perception of potential information system achievements and thus fail to service the real needs of management [Avison and Wood-Harper 1991]. As information technology migrates into increasingly more complex roles within organizations, designers must build systems that correctly address enigmatic yet essential organizational needs. Designers must first achieve a deep understanding of the essential application problem. The complexity of these new information problem arenas exceeds the boundary of physical partitioning. Abstract modeling appears to be the only means to understand the essentials of these problems and to convey these essentials from one phase of the project to another. That is, abstract models become the strong linkage between the requirements definition and the specification of design that was missing in the second generation. Ward and Mellor [1985] denominate this shift toward problem-dominated methods as one key generalization about the evolution of systems development methods. Friedman [1989, p. 174] characterizes this third phase as one in which "producing the right system, rather than producing the system right, became the primary criterion for a successful system."

Since there are widely varied perspectives on the nature of organizational problems, the nature of the abstract system models varies considerably. These models can be classified into two broad categories. *Logical* models tend to express system needs and behavior in a functional or data-oriented sense. They sometimes begin with a physical model of existing systems. References to physical processing technologies or data storage media are then removed to create the logical model. New system design transformations are merged into this logical model. The logical model is translated back into a physical model by establishing technological boundaries around the processing and data elements. *Transformational* models are a highly distinctive second class of system models. These models express organizational (not systems) needs and behavior in a social or theoretical sense and are characteristic of methods that look beyond the functional information requirements to focus on problem formulation, job satisfaction, and worker autonomy or emancipation. The term "transformational" denotes an emphasis on transformation in work force management [Bass 1985], organizational theory [Leifer 1989], and social philosophy as a whole [Lyotard 1987]. Transformational models tend to be less formal in nature because they must be highly idiographic to be effective in diverse organizational settings. They will usually focus on learning rather than analysis, and a participation or reflection stage will often establish and validate an implicit or explicit transformational model [Ehn 1989;

Golrang and Hägerfors 1989; Land 1982; Saarinen and Sääksjävi 1989; and Schön 1983].

Logical modeling methodologies for general information systems analysis and design include the structured software engineering techniques of Yourdon and Constantine [1979], the structured analysis and design techniques of DeMarco [1979], Gane and Sarson [1984], and Yourdon [1989], the data-modeling methods evolving from Chen [1976], information engineering from Finkelstein [1989] and Martin [1990], and most recently the object-oriented design methods of Coad and Yourdon [1991], Rumbaugh et al. [1991], and Embry et al. [1992]. Transformational modeling methodologies include participative sociotechnical methods, such as Mumford and Weir's [1979] ETHICS, the learning and reflection cycles characteristic in Checkland's [1981] Soft Systems Methodology, and the emphasis on ideal work life system "discovery" arising in cooperative prototyping [Gronbaek 1989].

This generation of methods deemphasizes cost-benefit analysis because the abstract modeling phase more closely links the system solution to the organizational problems. In logical models, this link is made in a highly functional sense. All system elements are clearly projected in their essential roles as part of the system solution. In transformational models, this link is made in a highly behavioral sense. The essential impact of all system elements becomes clearly understood. The explicit nature of the problem-solution links in these models severely limits the probability of unnecessary or redundant system elements in the final system solution. As a result, cost-benefit analysis is less important in determining the final selection of system elements.

3.2 Formative Logical Security Design Methods

We can extend third-generation systems development concepts to gain an understanding of the nature of logical transformational security methods. We should discover at least three distinguishing characteristics of third-generation security methods. First, there will be an emphasis on producing the right types of security for the system, not just implementing the security correctly. Second, the security design method will be characterized by either logical models or transformational models (or both). Finally, a deemphasis of the cost-benefit risk analysis will accompany the increased usage of abstract models.

These characteristics help us to recognize that the work in the third generation of security methods is formative. There are no complete examples of third-generation security design methods. Nevertheless, two methods have been published that approach these third-generation criteria: the CCTA SSADM-CRAMM interface and the Logical Controls Design method. Together, they provide a strong indication of the future direction of major developments in security controls design.

3.2.1 SSADM-CRAMM Interface

The CCTA is responsible for the U.K. government standard Structured Systems Analysis and Design Method (SSADM), as well as the CRAMM second-generation security method described earlier in Section 2.3.1. CCTA has extended the CRAMM method into the overall systems development process by developing an interface between CRAMM and SSADM [CCTA 1991]. CRAMM is the only second-generation method that has been rationalized into an overall information system development method.

SSADM is a third-generation systems development method that involves logical models in several stages. The left section of Figure 6 is an overview of version 3 of the SSADM method. Stages 01 and 02 comprise an optional feasibility study. Stage 1 is an analysis of system operations and current problems. Stage 2 is the specification of requirements. Stage 3 is the selection of technical options. Stages 4 and 5 are data and process de-

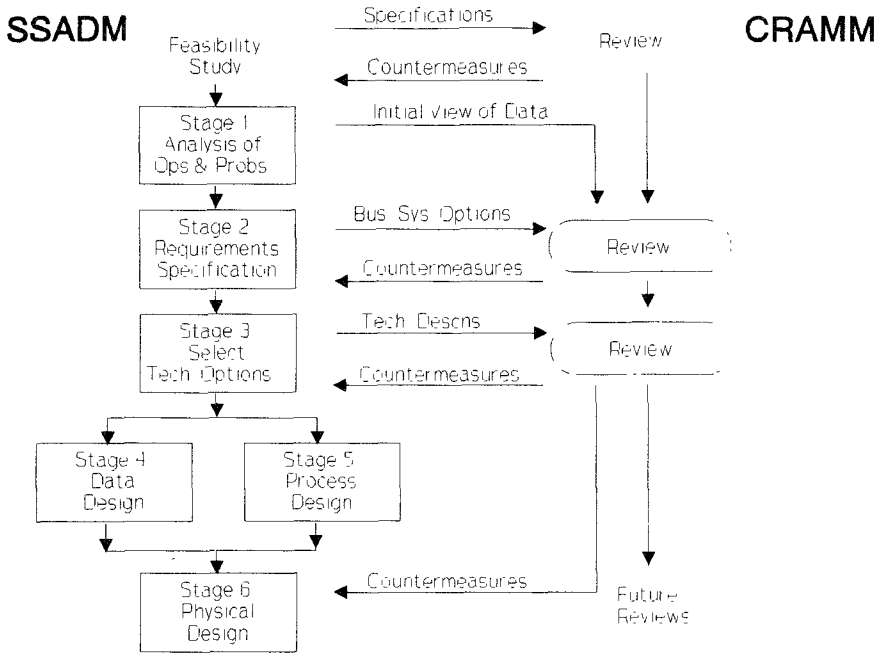


Figure 6. Overview of SSADM-CRAMM interface.

sign and may take place in parallel. Stage 6 is physical design. Logical modeling tools (e.g., dataflow diagrams) are introduced in stage 2 of SSADM. SSADM embodies the major characteristics of the third generation with its emphasis on business options, logical models, and curtailed concern for cost-benefit feasibility.

The right section of Figure 6 illustrates the CRAMM reviews and exchanges of SSADM and CRAMM results. The CRAMM reviews involve the three CRAMM stages explained in Section 2.3.1 above. Each CRAMM review examines the evolving SSADM design information. The project specifications from the feasibility study are used as the foundation of the first CRAMM review. The initial view of the data (where no feasibility study was conducted) and the business system options, together with the results of the first review, form the basis of the second CRAMM review. The technical environment descriptions in conjunction with the results from the second CRAMM review support the third CRAMM review. Each CRAMM review yields proposed countermeasures that are introduced back into the SSADM design decisions. The results of the final SSADM-CRAMM review are carried forward to routine operational CRAMM reviews that are part of the system management.

The CRAMM review does not operate directly on any of the logical models developed by SSADM. The interface recognizes that a CRAMM model with no physical assets will generate incomplete results. Consequently, the designers at each review must introduce broad assumptions about the physical assets expected for the proposed system to produce recommendations for countermeasures that can be compared realistically. In other words, CRAMM can only be used during the logical modeling phases of SSADM if the developers create an "assumed" physical model. The two methods continue to use separate system specifications and models. CRAMM itself remains intact as a second-generation method. A side effect of this development is the larger presence of risk analysis (through the role of CRAMM) in third-generation security methods when compared with more gen-

eral third-generation development methods and their dramatically subdued role for cost-benefit analyses. The SSADM-CRAMM interface creates a unique hybrid approach, however, that employs second-generation security design at certain points within a third-generation systems development method.

In this sense, CRAMM-SSADM is only formative as a third-generation method for security design. Nevertheless, it demonstrates that system developers are beginning to recognize the need for security considerations as part of the fundamental process of systems design. The status of both CRAMM and SSADM as U.K. government standards means that the CRAMM-SSADM hybrid is an important advance toward the third generation of security design.

3.2.2 Logical Controls Design Method

At least one method for security design has been published that matches the criteria for logical systems engineering. This "Logical Controls Design" method [Baskerville 1988; 1989] focuses the security design process away from the hardware. The focus shifts to the software and work procedures that access and manipulate information. The purpose of this shift is to emphasize logical controls that may be expected to endure longer than physical controls.

In this method, systems security analysis and design are carried out in parallel with DeMarco's [1979] structured systems analysis. First, designers create a physical model of the existing system; then they translate this physical model into a logical model; and last they introduce controls into the logical model. As a side benefit of this method, the lack of physical aspects of risks reduces the usual six classes of risk[9] to three logical classes of risk: destruction, modification, and disclosure.

All controls take the form of a process (or a part of a process) that may require certain additional data. For example, suppose the data structure under consideration is an order entry system CUSTOMER-ORDER dataflow. One risk is incorrect "modification" of the data. One type of detective control involves using batch totals. The batch total control would require a data element (the BATCH-TOTAL-AMOUNT) and a process (BALANCE-TO-CONTROLS). The analyst adds this control data element and control process element to the overall system logical model. Thus, the logical system model is also used as the means for describing the logical security model. Since this model will not normally provide any concise indication of the completeness of the security control set, however, the method adds cross-references in each data dictionary entry. This cross-reference lists both the threat classes and the processes that provide the controls for each threat class, thus documenting completeness of security design. The exact structures of the security process elements are then documented like any normal process. That is, the control process is an element in the overall dataflow diagram and a related entry in the functional primitive descriptions.

For example, the data dictionary entry for a file containing customer account numbers might contain the following cross-reference notes:

Modification Risk Control:
CHECK-DIGIT-VERIFICATION
Destruction Risk Control:
GENERATE-CUSTOMER-BACKUP-FILE
Disclosure Risk Control:
ENCRYPT-CREDIT-ENTRY

In this example, CHECK-DIGIT-VERIFICATION is a process documented elsewhere in the specification (as are the backup and encryption processes). The design of the controls and control data structures are detailed as other elements of the system under analysis. In this way, a logical view of the security design is achieved

[9] Six classes of risk are typical in much security methodology. They arise in a matrix that maps intentional and accidental motives onto destruction, modification, and disclosure (Table 6, Fisher's exposure groups).

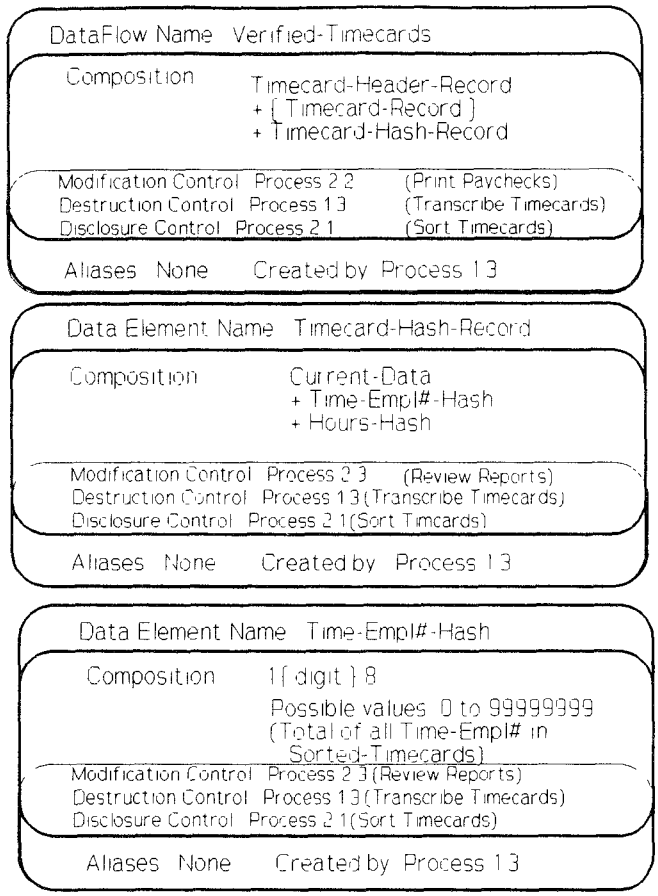


Figure 7. Logical Controls Design data dictionary entries with security.

that transcends physical limitations. Check digits, encryption, and backup offer protection against intentional or accidental modification. Figure 7 offers examples of such cross-reference entries in the data dictionary.

Controls for each process are designed along similar lines to the data elements described so far. Processes can be destroyed (resulting in the loss of a system "state"), disclosed (opening vulnerabilities in system control flows), or modified (impacting inputs and outputs, as well as data and control flows). The security designer documents the logical controls for process elements by cross-referencing the three relevant control processes in each process transform description. For example, a process that writes a purchase order might contain security documentation as follows:

WRITE-PURCHASE-ORDER

Modification Control:
APPROVE-PURCHASE-ORDER
Destruction Control:
LOG-PURCHASE-ORDER
Disclosure Control:
SEAL-PURCHASE-ORDER

The control processes APPROVE-PURCHASE-ORDER, LOG-PURCHASE-ORDER, and SEAL-PURCHASE-ORDER are detailed in transform descriptions elsewhere in the speci-

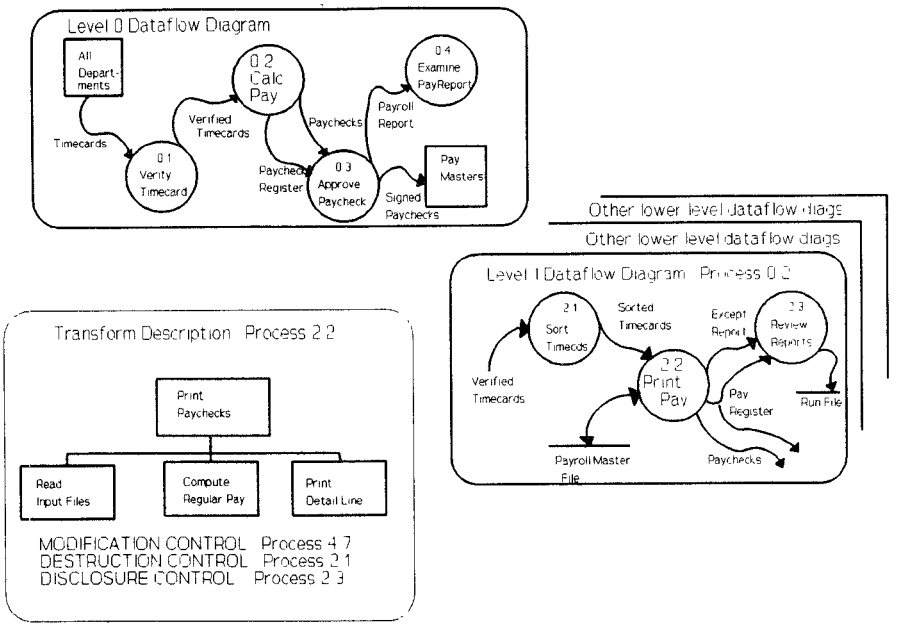


Figure 8. Logical Controls Design dataflow diagram with security.

fication. These control transformation steps detail mechanisms that prevent modification, provide failure restart, and correct for process disclosure, respectively. By including one or more control cross-references in each process transform description, the design is assured of a complete control set. Figure 8 is an example of a dataflow diagram with security.

This particular method is rare because it qualifies as a third-generation method for systems security design. It achieves a logical view of the risk/control design that transcends many physical complications (e.g., whether a modification was intentional or accidental). As a third-generation method, however, this work is only formative. It parallels the "classical" forms of logical transformational methods that predate fourth-generation languages, prototyping tools, personal computers, and CASE [Yourdon 1989]. The Logical Controls Design method is also limited because it employs two weakly connected and poorly developed transformational models. There is a simple model of the impact of controls as constraints on system performance and a simple stakeholder analysis that delineates people impacted by system security. Without a means to connect the three models, this method cannot properly address the security concerns captured in the actual work patterns of human behavior [Bannon 1989].

3.3 Assumptions and Activities of Third-Generation Security Methods

Five fundamental assumptions characterize the formative third generation. First, ideal security solutions will evolve only from an understanding of the broad problem situation. This assumption rejects the second-generation notion that the correct control set will be an obvious deduction following a study of the detailed requirements. The second assumption is that abstraction is the path to such an understanding. Abstract models clarify the organizational problems, and as a consequence more effective solutions

Table 11. General Characteristics of Third-Generation Security Methods

Third-Generation Logical-Transformational Methods			
Assumptions	**Activities**	**Strengths**	**Weaknesses**
▪ Ideal security solutions will evolve only from an understanding of the broad problem situation. ▪ Abstract models clarify the organizational problems, and more effective controls will result ▪ Designs founded on such models will prove flexible, adaptable and consequently longer-lived. ▪ There are few universal solutions ▪ Controls place constraints on information systems	▪ Model building ▪ Stakeholder analysis ▪ Translation of abstract models into reality ▪ Implementing physical models ▪ Maintaining	▪ Flexibility in controls ▪ Excellent documentation ▪ Low maintenance costs ▪ Closer connection ▪ Lessen the relevance of risk analysis ▪ A concerted security-functionality design process ▪ Less conflict between security and system usability	▪ Lack of experience ▪ Abstract controls are difficult ▪ Physical security details are postponed ▪ Difficult cost evaluation ▪ Not as useful for existing systems

will result. Third, control designs founded on such models will prove flexible and consequently longer lived. System controls derived in such models will be more general to the organizational situation and will be more adaptable in periods of unpredicted change. This regard for the organizational situation implies a fourth assumption, namely, that there are no universal solutions. In short, controls must be specifically designed for each problem system. Finally, the cost of controls is assumed to outweigh the benefits of controls in importance as a design issue. Controls place constraints on information systems. No constraint should ever be emplaced without close scrutiny of the impact on the system and, as a consequence, on the organization and its environment (Table 11).

The implications of these fundamental assumptions lead to five types of activities. The context includes the integration of the security design activities with other system design activities. Model building is the first type. It follows the assumption that a deep understanding of the broad problem situation is necessary and that abstract models will clarify organizational security problems. Model building also reflects the assumption that model-based designs yield flexible and maintainable system controls. The second type is stakeholder analysis, and it also results from an assumption that security designers need a broad problem understanding. Additionally, this type of activity also follows from the assumption that controls can constrain information system behavior and affect various stakeholders differently. A third type is translation of abstract models into physical models. Such translation activities are required to materialize the benefits of the abstract models into reality. Implementation of a physical model is a fourth kind of activity. While this type naturally follows the activity of physical modeling, it also reflects the assumption that few universal solutions exist. Implementation activities must create unique controls for each particular problem situation. The fifth type is maintenance of the models and the represented system. To a large degree, this type reflects the assumption that good models will produce adaptable, long-lived systems. The maintenance activities are needed to keep the security in concordance with the system functionality.

3.4 Strengths and Weaknesses of Third-Generation Methods

Unfortunately, without several examples of such methods, it is not possible to discuss the common strengths and weaknesses. The hybrid SSADM-CRAMM interface highlights the problem of mixing two generations of method: separate models and notation for functionality and security. This mix would invalidate any attempts to generalize the practical aspects of SSADM-CRAMM to an entire generation of methods. However, the Logical Controls Design method, as a formative exemplar, does imply several general strengths and weaknesses that we could expect from future third-generation security methods.

Early experience indicates that flexibility in the control structures is a major strength of third-generation methods. Much of the security is designed into an abstract model of the system. Removal of the design from the physical realm adds a degree of technological independence. The human-machine boundary can shift, or changes in the underlying technology can occur, without changing the essential nature of the security implementation. This flexibility, together with the excellent documentation produced in system models, suggests that the maintenance costs for system security will be reduced substantially. The abstract models also imply a closer connection between the essential organizational problems and the security design. This close connection will lessen the relevance of risk analysis (and its problematical elements) since few controls will be specified that are not directly linked to system functions. Also, other third-generation methods might attach the security model to the more general abstract system model (as in the Logical Controls Design method). This regard for organizational problems orients the design process toward harmonized security and functionality. It eliminates the separate uncoordinated independent designs. As a result, less conflict should exist between security and system usability.

Lack of experience with third-generation methods will mean designers must struggle to invent abstract controls for use in abstract models. Many current controls depend on a particular technological foundation (e.g., encryption, dialup access control, and file locking), and these controls cannot be described easily in an abstract model. Third-generation techniques also share a weakness when it comes time to migrate from the abstract models toward a real working system. Many implementation details are postponed until late in the design process. Physical security can probably never be eliminated completely, and it is not clear how third-generation methods will resolve or integrate both logical and physical security components. It appears that third-generation methods, by integrating the system and security designs, will obscure the costs of security controls by merging these costs with other functional elements. A separate cost-benefit evaluation for security will become a problem. Finally, it appears likely that third-generation methods will tend to rely heavily on an abstract modeling technique obtained from a more general third-generation system design method. This reliance implies these methods will be oriented toward adding security to new systems. They will not be as useful in formulating security for the large population of existing computer-based information systems.

3.5 Research Directions for Third-Generation Methods

If we consider the third-generation work in light of the other generations, a future research arena appears that has important consequences for general information systems development, as well as security development. This arena emerges only when we closely consider two aspects of the relationship between security methods and general systems methods. The first aspect regards the discord between current research into security development and the present work in

general IS development. The focus of the general work has rested in the third generation for at least a decade. The research into third-generation security methods, however, is incipient. Most of the current interest in security methods still focuses on the second generation.

This discord between the work in general methods and the work in security methods suggests that security methodologies are underdeveloped. Security work seems to lag the general IS development research arena. The discord further suggests that security methodologists might aim toward development of new logical transformational security methods based on the knowledge developed in the general arena.

These suggestions may seem preliminary, however, when we consider a second aspect of the relationship between security methods and general IS development methods. This aspect regards the relationship between practice and third-generation methods. Authorities dispute whether practitioners have ever been able to use the widely published third-generation IS development methods. Various empirical studies claim that current system developers have substantially adopted third-generation methods (e.g., Necco et al. [1987], Necco [1989], Saarinen [1990], and Sumner [1992]). Other studies offer evidence that such espoused methods are only nominal, however, and that actual work processes differ substantially from the claimed method (e.g., Parnas and Clements [1986], Gause and Weinberg [1989], Baskerville et al. [1992], and Bansler and Bødker [1993]). These challenges claim that practitioners broadly aspire to third-generation methods, but they cannot achieve their use in practice.

The first aspect above (discord) indicates that some practical problem may be interfering with research into third-generation security methods. The second aspect (practice) indicates that there may be a fundamental practical defect in general third-generation IS development methods. Perhaps these two phenomena are related and appear as consequences of a single defect in our thinking about methods. This single defect could be the lack of security considerations in general IS development methods. That is, current third-generation IS development methods may be impractical because they fail to guide the practitioner to a safe system. Practitioners must intuitively deviate from development methods to incorporate threat-reducing strategies. In this sense, most widely known third-generation IS development methods are incomplete.

The concept of an integrated system development and security method is an important one. It has developed gradually across the generations. In the first generation there was no integration. In the second generation it appeared as an option for some techniques (e.g., control flowcharts in second-generation system flow diagrams [Jenkins and Carlis 1988]). In the third generation, the formative techniques depend completely on elements of general IS development methods.

This is a very strong direction for future research into third-generation methods. The complete integration of security considerations with a more general IS development method may yield an approach that could gain more than just nominal practice. Practitioners could use such an approach to express more effectively their concerns for both functionality and safety in future systems.

The discord aspect alone suggests expansive new third-generation security work. Contrast this with the implication of the two joint aspects: elimination of independent security methods altogether. If both IS development and security methods must be integrated to succeed in the third generation, the second generation may culminate practical independent security methods forever. Further independent security methodologies may not be able to advance into the third generation much further than the two existing formative studies. Techniques arising from such work may prove increasingly unproductive in practice [Baskerville 1993].

4. CONCLUSIONS

Security is an important component of information systems. Security analysis and design methods have evolved in a manner similar to general IS development methods. They share such common traits as objectives, means, challenges, and primary concepts. The earliest security methods focused on checklists and simple risk analysis for decision support. Later methods focus on the mechanistic partitioning of complexity in a desired system. They entail a search for those critical controls that provide minimum satisfactory protection for an entire information system. More recently, interest is developing in methods that focus on abstract models as a means for understanding the diverse security needs in an information system.

After many years of intellectual progress, security methods may have reached an important juncture with more general IS development methods. Security methods seem unable to make much progress beyond this juncture without their integration into these more general methods. Indeed, the general methods may be unable to gain practical success beyond this juncture without providing such express means for analyzing and designing security controls. An important synergy between two streams of research is one likely outcome. This outcome is a concern for security and safety in systems analysis that reflects the merger of security methodology with mainstream systems development methodologies.

ACKNOWLEDGMENTS

The author gratefully acknowledges the continued encouragement of Heinz K. Klein and the comments of Rodney Clark, Will Ozier, and several insightful referees. Thanks also go to Michael Davidson for his bibliographic assistance.

REFERENCES

AGRESTI, W. 1986. What are the new paradigms. In *New Paradigms for Software Development*. IEEE Press, Washington, D.C., 6–10.

AMERICAN BAR ASSOCIATION 1984. *Report on Computer Crime*. American Bar Ass., Section on Criminal Justice, Task Force on Computer Crime, Washington, D.C.

AVISON, D., AND FITZGERALD, G. 1988. *Information Systems Development: Methodologies, Techniques and Tools*. Blackwell Scientific, Oxford, U.K.

AVISON, D., AND WOOD-HARPER, T. 1991. Information systems development research: An exploration of ideas in practice. *Comput. J. 34*, 2, 98–112.

BADENHORST, K., AND ELOFF, J. 1990. Computer security methodology: Risk analysis and project definition. *Comput. Sec. 9*, 4 (June), 339–346.

BANNON, L. 1989. Discussant notes on Baskerville and Hellman. In *Systems Development for Human Progress*. North-Holland, Amsterdam, 257–260.

BANSLER, J., AND BØDKER, K. 1993. A reappraisal of structured analysis: Design in an organizational context. *ACM Trans. Inf. Syst. 11*, 2, 165–193.

BASKERVILLE, R. 1993. The threat in security for the adaptive organization. *Inf. Syst. Sec 2*, 1 (Spring), 40–47.

BASKERVILLE, R. 1992. The developmental duality of information systems security. *J. Manage. Syst 4*, 1, 1–12.

BASKERVILLE, R. 1991. Risk analysis as a source of professional knowledge. *Comput. Sec. 10*, 8 (Dec.), 749–764.

BASKERVILLE, R. 1989. Logical controls specification: An approach to information systems security. In *Systems Development for Human Progress*. North-Holland, Amsterdam, 241–256.

BASKERVILLE, R. 1988. *Designing Information Systems Security*. Wiley, Chichester, U.K.

BASKERVILLE, R., TRAVIS, J., AND TRUEX, D. 1992. Systems without method. In *IFIP Transactions on The Impact of Computer Supported Technologies on Information Systems Development*. North-Holland, Amsterdam, 241–270.

BASS, B. 1985. *Leadership and Performance Beyond Expectation*. Free Press, New York.

BLOOMBECKER, B. 1990. *Spectacular Computer Crimes: What They Are And How They Cost American Business Half A Billion Dollars A Year*. Dow Jones-Irwin, Homewood, Ill.

BROWNE, P. 1979. *Security: Checklist for Computer Center Self-Audits*. AFIPS Press, Arlington, Va.

BUI, T., AND SIVASANKARAN, T. 1987. Cost-effectiveness modeling for a decision support system in computer security. *Comput. Sec. 6*, 2, 139–151.

CARROLL, J., AND MACIVER, W. 1984. Towards an expert system for computer facility certification. In *Computer Security: A Global Challenge* North-Holland, Amsterdam, 293–306.

CCTA 1991. *SSADM-CRAMM Subject Guide for SSADM Version 3 and CRAMM Version 2* Central Computer and Telecommunications

Agency, IT Security and Privacy Group, Her Majesty's Government, London.

CHECKLAND, P. 1981 *Systems Theory, Systems Practice*. Wiley, Chichester, U.K

CHEN, P. 1976. The entity-relationship model: Toward a unified view of data. *ACM Trans Database Syst. 1*, 1 (Mar.), 9–36

CLEMENTS, D. 1977. Fuzzy models for computer security system metrics Ph.D. thesis, Dept. of Electrical Engineering and Computer Sciences, Univ of California at Berkeley, Berkeley, Calif

COAD, P., AND YOURDON, E. 1991. *Object-Oriented Analysis* 2d ed. Yourdon Press, Englewood Cliffs, N.J.

COMMISSION OF EUROPEAN COMMUNITIES 1990. *Information Technology Security Evaluation Criteria (ITSEC), Provisional Harmonized Criteria, Version 1.2* Commission of European Communities, Directorate—General XIII, Brussels, Belgium.

COMMUNICATIONS SECURITY ESTABLISHMENT 1990. *Automated Risk Analysis Product Assessment*. Canadian System Security Center, Government of Canada, Ottawa.

COMPUTER SECURITY CONSULTANTS 1988. *Using Decision Analysis to Estimate Computer Security Risk*. Computer Security Consultants, Ridgefield, Conn.

COMPUTERWORLD 1983. Computer crime in Japan. *Computerworld 17*, 45 (Nov. 7), ID7–ID8, ID17–ID20.

COUGER, J. 1982. Evolution of system development techniques. In *Advanced System Development/Feasibility Techniques*. Wiley, New York, 6–13

COURTNEY, R. 1977. Security risk assessment in electronic data processing. In the *AFIPS Conference Proceedings of the National Computer Conference 46*. AFIPS, Arlington, Va., 97–104.

DAVIS, G. 1982. Strategies for information requirements determination *IBM Syst. J 21*, 1, 4–30.

DEMARCO, T. 1979. *Structured Analysis and System Specification* Yourdon Press, New York.

DIXON, R , MARSTON, C., AND COLLIER, P 1992. A report on the joint CIMA and IIA computer fraud survey. *Comput Sec. 11*, 4 (July), 307–313.

EHN, P 1989 The art and science of designing computer artifacts. *Scand. J. Inf. Syst. 1*, (Aug), 21–42

EMBRY, D., KURTZ, B., AND WOODFIELD, S. 1992. *Object-Oriented Systems Analysis. A Model-Driven Approach* Yourdon Press, Englewood Cliffs, N.J.

FARQUHAR, B. 1991. One approach to risk assessment. *Comput. Sec 10*, 1, 21–23

FINKELSTEIN, C. 1989. *An Introduction to Information Engineering. From Strategic Planning to Information Systems*. Addison-Wesley, Sydney, Australia.

FISHER, R 1984. *Information Systems Security*. Prentice-Hall, Englewood Cliffs, N J

FITES, P., JOHNSTON, P., AND KARTZ, M. 1989 *The Computer Virus Crisis*. Van Nostrand Reinhold, New York.

FITZGERALD, J. 1978 *Internal Controls for Computerized Systems*. Underwood Press, San Ceandro, Calif.

FITZGERALD, J., AND FITZGERALD, A. 1990 *Designing Controls into Computerized Systems*. 2d ed. Jerry FitzGerald & Associates, Redwood City, Calif

FRIEDMAN, A. 1989 *Computer Systems Development: History, Organization and Implementation*. Wiley, Chichester, U K.

GALLEGOS, F., RICHARDSON, D , AND BORTHICK, A. 1987. *Audit and Control of Information Systems*. South-Western, Cincinnati, Ohio.

GANE, C., AND SARSON, T. 1984. *Structured Systems Analysis· Tools and Techniques* Prentice-Hall, Englewood Cliffs, N.J.

GANNON, P 1992. French losses rise sharply. *Comput. Fraud Sec Bull. 14*, 12 (Oct), 3.

GAUSE, D., AND WEINBERG, G. 1989. *Exploring Requirements: Quality Before Design*. Dorset House, New York.

GILBERT, I 1989. *Guide for Selecting Automated Risk Analysis Tools* U.S Department of Commerce, National Institute of Standards and Technology, NIST special publication 500-174 (Oct.), Washington, D.C.

GLASEMAN, S., TURN, R., AND GAINES, R. 1977 Problem areas in computer security assessment. In *Proceedings of The National Computer Conference NCC 46*. AFIPS Press, Arlington, Va , 105–112

GOLRANG, T., AND HAGERFORS, A. 1989. It's like walking in syrup—a participative change process. In *Proceedings of the 12th IRIS—Part I*. Computer Science Dept., Aarhus Univ., DAIMI PB 296-I, Aarhus, Denmark, 183–202

GRONBAEK, K. 1989. Extending the boundaries of prototyping: Towards cooperative prototyping. In *Proceedings of the 12th IRIS*. Aarhus Univ., DAIMI PB 296-I, Aarhus, Denmark, 219–238.

GUARRO, S. 1987 Principles and procedures of the LRAM approach to information systems risk analysis and management *Comput Sec. 6*, 6, 493–504.

HAFNER, K., AND MARKOFF, J. 1991 *Cyberpunk: Outlaws and Hackers on the Computer Frontier*. Simon and Schuster, New York

HAWRYSZKIEWYCZ, I. 1988. *Introduction to Systems Analysis and Design* Prentice-Hall, Englewood Cliffs, N.J

HEMPHILL, C., AND HEMPHILL, J 1973 *Security Procedures for Computer Systems* Dow Jones-Irwin, Homewood, Ill

HIGHLAND, H 1992 Random bits and bytes: Michelangelo—Part II. *Comput. Sec. 11*, 4 (July), 294–303.

HIRSCHHEIM, R., AND KLEIN, H. 1992. Paradigmatic influences on information systems development methodologies: Evolution and conceptual advances. *Adv. Comput. 34*, 294–381.

HOFFER, J., AND STRAUB, D. 1989. The 9 to 5 underground: Are you policing computer crimes? *Sloan Manage. Rev. 30*, 4 (Summer), 35–43.

HOFFMAN, L., MICHELMAN, E., AND CLEMENTS, D. 1978. SECURATE—security evaluation and analysis using fuzzy metrics. In *AFIPS National Computer Conference Proceedings 47*. AFIPS, Arlington, Va., 531–540.

HOYT, D. 1973. *Computer Security Handbook*. Macmillan, New York.

HRUSKA, J. 1990. *Computer Viruses and Anti-Virus Warfare*. Ellis Horwood, New York.

HUTT, A., BOSWORTH, S., AND HOYT, D., EDS. 1988. *Computer Security Handbook*. 2d ed. Macmillan, New York.

IBM 1972a. *Secure Automated Facilities Environment Study 3*. Part 2 (May). IBM, Armonk, N.Y.

IBM 1972b. *DP Asset Protection Self-Assessment Guide*. Reprinted in *Information Systems Security*. Prentice-Hall, Englewood Cliffs, N.J., 1984, 212–231.

JENKINS, A M., AND CARLIS, J 1988. Control flowcharting for data driven systems. *Informatica 2*, 76–82.

KRAUSS, L. 1980. *SAFE: Security Audit and Field Evaluation for Computer Facilities and Information*. Revised ed. Amacon, New York.

KRAUSS, L. 1972. *SAFE: Security Audit and Field Evaluation for Computer Facilities and Information Systems*. Amacon, New York.

LAND, F. 1982. Notes on participation. *Comput. J. 25*, 2 (May), 283–285.

LANDRETH, B. 1989. *Out of The Inner Circle: The True Story of A Computer Intruder Capable of Cracking The Nation's Most Secure Computer Systems*. Tempus, Redmond, Wash.

LANDWEHR, C. E. 1981. Formal models for computer security. *ACM Comput. Surv. 13*, 3 (Sept.), 247–278.

LEIFER, R. 1989. Understanding organizational transformation using a dissipative structure model. *Hum. Rel. 42*, 10, 899–916.

LUCAS, H. 1976. *The Analysis Design and Implementation of Information Systems*. McGraw-Hill Kogakusha, Tokyo.

LYOTARD, J-F. 1987. The postmodern condition. In *After Philosophy: End or Transformation*. MIT Press, Cambridge, Mass., 73–93.

MAIR, W., WOOD, W., AND DAVIS, K. 1978. *Computer Control and Audit*. Prentice-Hall, Englewood Cliffs, N.J.

MARTIN, J. 1990. *Information Engineering*. Books I–IV. Prentice-Hall, Englewood Cliffs, N.J.

MARTIN, J. 1973. *Security, Accuracy and Privacy in Computer Systems*. Prentice-Hall, Englewood Cliffs, N.J.

MCLEAN, J. 1990. The specification and modeling of computer security. *Computer 23*, 1 (Jan.), 9–16.

MUMFORD, E., AND WEIR, M. 1979. *Computer Systems in Work Design: The ETHICS Method*. Associated Business Press, London.

MURDICK, R. 1980. *MIS Concepts and Design*. Prentice-Hall, Englewood Cliffs, N.J.

NATIONAL RESEARCH COUNCIL 1991. *Computers At Risk: Safe Computing in the Information Age*. National Academy Press, Washington, D.C.

NECCO, C. 1989. Evaluating methods of systems development: A management survey. *J. Inf. Syst. Manage. 6*, 1 (Winter), 8–16.

NECCO, C., GORDON, C., AND TSAI, N. 1987. Systems analysis and design: Current practices. *MIS Q. 11*, 4 (Dec.), 461–476.

NIELSEN, N., AND RUDER, B. 1980. Computer system integrity vulnerability. *Inf. Privacy 2*, 1 (Jan.), 21–25.

NOLAN, R. 1979. Managing the crisis in data processing. *Harvard Bus. Rev. 57*, 2 (Mar.–Apr.), 115–126.

NORDBOTTEN, J. 1985. *The Analysis and Design of Computer-Based Information Systems*. Houghton Mifflin, Boston.

OZIER, W. 1992. Risk assessment and management. In *Data Security Management*. Report 85-01-20. Auerbach, New York.

OZIER, W. 1989. Risk quantification problems and Bayesian Decision Support System solutions. *Inf. Age 11*, 4 (Oct.), 229–234.

PARKER, D. 1986. *Computer Crime: Computer Security Techniques*. U.S Department of Justice, Bureau of Justice Statistics Document J29.2:C86, Washington, D.C.

PARKER, D. 1981. *Computer Security Management*. Reston, Reston, Mass

PARKER, D. 1976. *Crime by Computer*. Chas Scribners Sons, New York.

PARNAS, D., AND CLEMENTS, P 1986. A rational design process: How and why to fake it. *IEEE Trans. Softw. Eng. SE 12*, 2 (Feb.), 251–257.

PATRICK, B. 1974. Book review of SAFE. *Datamation 20*, 7 (Apr.), 208–209.

RUMBAUGH, J., BLAHA, M., PREMERLANI, W., EDDY, F., AND LORENSEN, W. 1991. *Object-Oriented Modeling and Design*. Prentice-Hall, Englewood Cliffs, N.J.

SAARI, J. 1991. Top management challenge: From quantitative guesses to prudent baseline of security. In *Proceedings of the 1991 IFIP Computer Security Conference* (Brighton, England, May). IFIP, Geneva, Switzerland, 295–300.

SAARI, J. 1987. Computer crime: Numbers lie. *Comput. Sec. 6*, 2, 111–117.

SAARINEN, T. 1990. System development methodology and project success: An assessment of situational approaches. *Inf. Manage. 19*, 3 (Oct.), 183–193.

SAARINEN, T., AND SÄÄKSJAVI, M. 1989 The missing concepts of user participation: An empirical assessment of user participation and information system success. In *Proceedings of the 12th IRIS—Part II* Computer Science Department, Aarhus Univ., DAIMI PB 296-II (Dec.), Aarhus, Denmark. 533–551

SALTMARSH, T., AND BROWNE, P. 1983 Data processing—risk assessment. In *Advances in Computer Security Management 2.* Wiley, Chichester, U.K., 93–116.

SCHÖN, D. 1983. *The Reflective Practitioner: How Professionals Think in Action.* Basic, New York.

SHELLY, G , AND CASHMAN, T. 1975 *Business Systems Analysis and Design.* Fullerton, Anaheim, Calif.

SMITH, S., AND LIM, J. 1984. An automated method for assessing the effectiveness of computer security safeguards. In *Computer Security A Global Challenge.* North-Holland, Amsterdam, 321–328.

SOLARZ, A. 1987. Computer-related embezzlement *Comput Sec.* 6, 1, 49–53

SPAFFORD, E. 1989 The internet worm: Crisis and aftermath *Commun. ACM 32*, 6 (June), 678–687.

STAMPER, R 1979. Lecture notes in systems analysis methodology 1 London School of Economics, London, U K

STOLL, C. 1989. *The Cuckoo's Egg Tracking a Spy through the Maze of Computer Espionage* Doubleday, New York.

SUMNER, M 1992. The impact of computer assisted software engineering on systems development. In *IFIP Transactions A8 the Impact of Computer Supported Technologies on Information Systems Development.* North-Holland, Amsterdam, 43–60.

U.S. DEPARTMENT OF COMMERCE 1979. *Guideline for Automatic Data Processing Risk Analysis.* Federal Information Processing Standards Publication FIPS 65 (Aug.), U.S. Dept. of Commerce, National Bureau of Standards, Washington, D.C.

U.S. DEPARTMENT OF DEFENSE 1985. *Trusted Computer Systems Evaluation Criteria* DOD 5200.28-STD. U S Dept. of Defense (Dec.), Washington, D C.

WARD, P., AND MELLOR, S. 1985. *Structured Development for Real-Time Systems* vol. 1 *Introduction and Tools.* Yourdon, Englewood Cliffs, N.J

WATERS, S. 1973 *Introduction to Computer Systems* NCC Publications, Manchester, U.K.

WEBER, R. 1988. *EDP Auditing Conceptual Foundations and Practice* 2d ed McGraw-Hill, New York

WHITESIDE, T. 1978. *Computer Capers Tales of Electronic Thievery, Embezzlement, and Fraud* Fitzhenry and Whiteside, Toronto.

WONG, K 1977. *Risk Analysis and Control.* National Computer Center Publications, Manchester, U.K.

WOOD, C 1990 Principles of secure information systems design. *Comput Sec 9*, 1 (Feb.), 13–24

YOURDON, E 1989. *Modern Structured Analysis.* Yourdon, Englewood Cliffs, N J.

YOURDON, E., AND CONSTANTINE, L. 1979. *Structured Design.* Prentice-Hall, Englewood Cliffs, N.J

ZVIRAN, M., HOGE, J., AND MICUCCI, V. 1990. SPAN–a DSS for security plan analysis. *Comput Sec 9*, 2, 153–160

Received May 1992, final revision accepted July 1993