CONF-9505260--1
SAND95-1558C

Managing Risk in Software Systems

Corresponding Author:
Dr. Sharon K. Fletcher
Sandia National Laboratories
MS0777
P. O. Box 5800
Albuquerque, NM, 87185-0777, USA

phone: 505/844-2251 // fax: 505/844-9641 // email: skfletc@sandia.gov

Additional Authors:
R. M. Jansma, Sandia National Laboratories, MS0484, Albuquerque, NM, 87185-0484
phone: 505/845-8254 // fax: 505/844-9524 // email: rmjansm@sandia.gov

M. D. Murphy, Sandia National Laboratories, MS0777, Albuquerque, NM, 87185-0777
phone: 505/844-5168 // fax: 505/844-9641 // email: mdmurph@sandia.gov

Dr. G. D. Wyss, Sandia National Laboratories, MS0747, Albuquerque, NM, 87185-0747
phone: 505/844-5893 // fax: 505/844-3321 // email: gdwyss@sandia.gov

J. Lim, Sandia National Laboratories, MS9011, Livermore, CA, 94550-0969
phone: 510/294-2973 // fax: 510/294-1225 // email: jlim@sandia.gov

Abstract
A methodology for risk management in the design of software systems is presented. It spans security, safety, and correct operation of software within the context of its environment, and produces a risk analysis and documented risk management strategy. It is designed to be iteratively applied, to attain appropriate levels of detail throughout the analysis. The methodology and supporting tools are discussed. The methodology is critiqued relative to other research in the field. Some sample applications of the methodology are presented.

Keywords: information systems, risk assessment, risk management, software surety, risk-based design

This work was supported by the United States Department of Energy under Contract DE-AC04-94AL85000.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

Managing Risk in Software Systems

Abstract
A methodology for risk management in the design of software systems is presented. It spans security, safety, and correct operation of software within the context of its environment, and produces a risk analysis and documented risk management strategy. It is designed to be iteratively applied, to attain appropriate levels of detail throughout the analysis. The methodology and supporting tools are discussed. The methodology is critiqued relative to other research in the field. Some sample applications of the methodology are presented.

Keywords: information systems, risk assessment, risk management, software surety, risk-based design

Introduction

Previous papers at this conference have included some excellent examples, from all over the world, of risk analyses which have been performed for real systems, from cellular phones to banking systems. [5, 6] The authors and analysts are to be commended for their serious efforts in identifying and mitigating system risks up front. In sharing their experiences, the authors pointed out difficulties such as: convincing auditors that old controls don't apply to new systems, reducing redundant controls, trading off privacy and data collection, and the need to compare vastly different approaches. And, while they seemed to do a complete job, the analyses were apparently ad-hoc, and therefore difficult to uphold. Clearly, efforts such as these could benefit from:

- more structured exploration of the risk space
- documentation of risk reduction objectives
- separation of objectives from how to achieve them
- what-iffing & understanding interactions
- demonstration & documentation of risk-related design decisions

Another previous paper at this conference [2] called for a Risk-Based Design Paradigm with the following building blocks:

1. Quantification of risk
2. Integrated requirements
3. Decision support

This aligns very well with what is needed to provide the benefits desired above. An even broader view would encompass not only risk-managed design, but also operation and maintenance of the system. The same three building blocks would apply, as long as they were designed broadly enough to encompass all lifecycle phases. We have made progress toward these building blocks in this broader context. This paper explains the risk management methodology we have derived, and gives examples of its application.

The Methodology

Requirements on the Methodology

The methodology must operate from a perspective which facilitates total risk management. It needs to support risk-based design activities and the development of a risk-management strategy for a system. A designer needs to be able to identify risk issues for a system over its entire lifecycle, and including dynamic aspects such as transitioning among operation, maintenance, and planned or unplanned shutdown. The designer must also be able to explore tradeoffs among design alternatives, and to demonstrate how a proposed design mitigates surety risks.

The problem has been cast in terms of surety risks, where surety is defined to encompass security, safety, dependability, etc. -- all the desirable attributes of a system in addition to its functional requirements. Total risk management means striving for correct system operation through appropriate levels of utility, integrity, access control, availability, and safety. Every system has its own unique surety issues, threats, and needs for risk reduction; thus every system needs a tailored risk analysis.

The methodology needs to quantify risk in a meaningful way and to produce output that is useful for decision making.

A Modeling Perspective

The perspective adopted here considers risk states (such as "data integrity lost", "incorrect output", "system unavailable", etc.) and how such states might be reached given various starting conditions (normal operation, maintenance, etc.). The heart of the methodology is a system risk model which depicts potential transitions between system risk states. Once the system is modeled from this viewpoint, barriers or risk mitigators can be inserted into the model and their effectiveness can be estimated. While this type of modeling is common in other fields using probabilistic risk assessment, it is not yet common in the software field. Many adaptations are required to deal with the character of software surety risks, the large uncertainties in quantified data, and the desired kinds of decision support.

The model must support a very broad interpretation of barriers, from software features, to physical protections, to operational procedures, to software development methodologies, to design techniques. These are all things that can mitigate surety risks. The modeling technique must allow barriers and threats to exert multiple influences throughout the system, so that an analyst can deal realistically with complexities such as: conflicts

between different surety objectives, secondary effects, multiple uses of a single barrier, multiple barriers to a single risk, etc. The modeling technique must also allow one to define threat agents and to move them through the system in such a way that the model reacts to the progress of the agent. For example, a threat agent may have certain resources and motivations which get "used up" as it traverses the system, and at the same time certain characteristics of the system may be irreparably changed by the agent.

Probabilistic risk assessment has been used for many years in the nuclear power, chemical, and other industries. Over time, data can be collected and corroborated for use in these assessments, although uncertainty of the data is always a concern. For the software industry, almost no data of the kind needed for risk assessments even exists. Thus, the techniques must be useful in a qualitative sense in the beginning, with the ability to handle a mix of qualitative and quantitative input as more data becomes available over time. Uncertainty, or confidence, in the data must be dealt with explicitly in the analytical techniques. Any given analysis is likely to consist of input that varies widely in both its granularity (qualitative-quantitative) and its uncertainty.

Recalling that the objective of the analysis suggested here is risk management, attention must be given to the form of the output. Reducing the output to some single risk number, for example, is of little use in either improving the system or documenting a risk management strategy.

Overview of the Risk Assessment Process

The system risk model provides a graphical depiction of potential system states and the barriers that can affect the probabilities of state transitions. But, more than that, it is also the formal description over which risk calculations can be defined. Thus, it is the heart of the analysis method. However, software system designers and analysts might not be familiar with this kind of modeling, and might have difficulty constructing meaningful and complete models. Thus, two matrices have been introduced as an aid to the model building. The **risk identification matrix** provides a framework within which the analyst can define perceived risk and desired risk reduction. The **risk mitigators matrix** provides information on potential barriers and their nominal effectiveness.

The overall **process**, shown in Figure 1, is for an analyst to **build a system risk model**, using the risk identification and risk mitigators matrices as guides and sources of information. Then the analyst performs a **barrier analysis** for each barrier, and a **threat analysis** for each threat, that is to be considered. Then an **analysis engine** is run to **evaluate remaining risk** in the system. The job of the analysis engine is to perform appropriate computations on all quantified input from the analyst, and to return information on weaknesses in the system. If cost information is incorporated, then the analysis engine returns **cost/benefit information** as well. The result of this process is a risk assessment and a risk management strategy for the system.

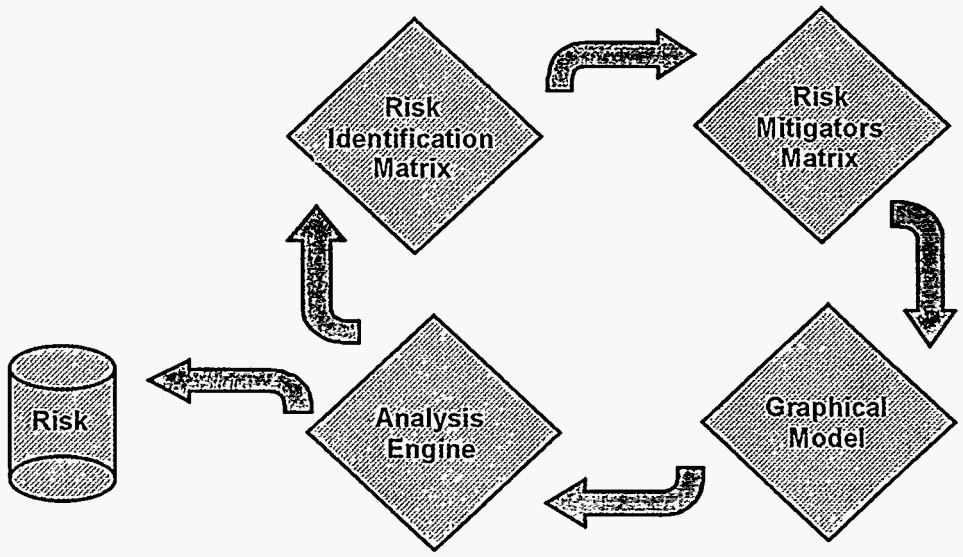

Figure 1. The Risk Assessment Process

Details of the Methodology

This section provides more detail on the following components of the methodology:

- risk identification matrix
- risk mitigators matrix
- system risk model
- process:
    - building the model
    - barrier analysis
    - threat analysis
    - analysis engine
        - risk evaluation
        - cost/benefit evaluation

The risk identification matrix, Figure 2, provides a hierarchy of risk sources for software-based systems. The rows of the matrix represent "Surety Objectives", and the columns represent "Aspects" of a system which might give rise to risks. The cells of the matrix contain sources of risk. The intent is for each cell to contain, ideally, all possible relevant sources of risk for any system, arranged as pieces of a hierarchy.

The matrix is read:
"There is a [surety objective] risk relative to [system aspect] due to [risk]."

Examples:

* "There is an [access control] risk relative to [system composition : network] due to [passwords exposed on network]."
* "There is an [integrity] risk relative to [information] due to [processing error]."
* "There is a [utility] risk relative to [state changes : shutdown] due to [shutdown-startup not synchronized]."
* "There is an [availability] risk relative to [processes] due to [system overload]."
* "There is a [safety] risk relative to [interfaces] due to [unchecked input]."

Although a more traditional view of impacts-assets [1] is accommodated within this framework, it is much broader, giving rise to exploration of system dynamics (State Changes), architecture choices (Composition), and correct operation (Utility). The intended purpose of the matrix is to guide the analyst's thinking into all relevant areas of risk and to suggest, but not limit, risks that should be considered. For a particular system undergoing analysis, the risk identification matrix will be both pruned and extended by the analyst to contain and prioritize only those risks of sufficient consequence and likelihood that they need to be mitigated. Consequences that should be considered include mission-related, political/social, health & safety, environmental, and regulatory/legal; these form a third dimension on the matrix.

	Information	Processes/ Transactions	System Composition hw-sw-nw-usr	State Changes op-mt-sh-ae	Interfaces
Access Control			passwords exposed on network		
Integrity	- intruder alters - processing error - user alters				
Utility				shutdown-startup not synchronized	
Availability		system overload			
Safety					unchecked input

hw = hardware
sw = software
nw = network
usr = user/operator

op = operational
mt = maintenance
sh = shutdown
ae = abnormal event

Figure 2. Risk Identification Matrix

The risk mitigators matrix has the same format as the risk identification matrix, but contains mitigators corresponding to risks. The intent is that the mitigators not be limited to hardware and software technologies, but include rules and procedures, design and development practices, and cover the lifecycle spectrum. Thus credit can be given for using a proven real-time design architecture, for using a highly rated software development methodology, for a trusted path delivery mechanism, for a fail-safe design, and so on.

Figure 3 illustrates one way of diagramming the system risk model.

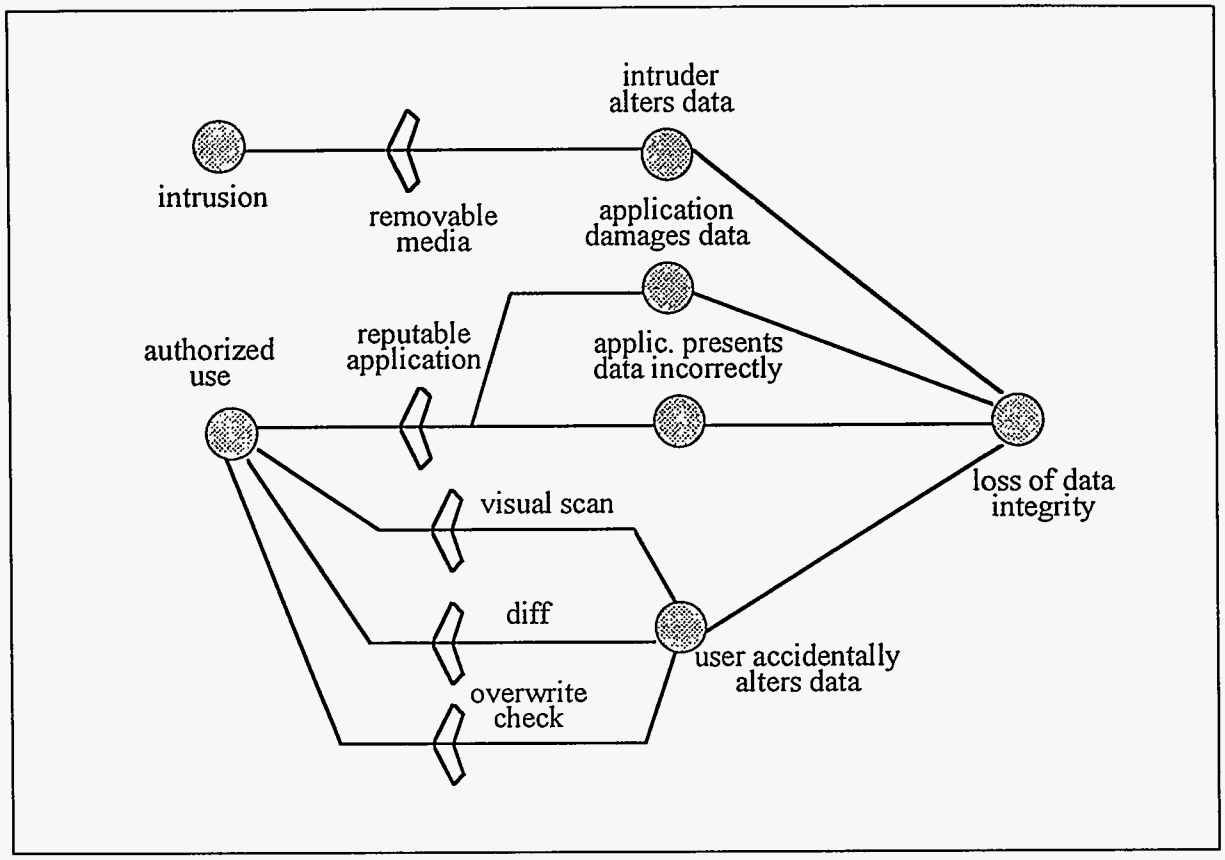

Figure 3. Graphical System Risk Model

Elements of the model are system states, represented by circles; transitions, represented by lines between circles; and risk mitigators, represented by the barrier symbol along transitions. The example in Figure 3 illustrates how one mitigator (use a reputable application) can mitigate two transitions, and how a variety of mitigators (visual scan, diff, overwrite check) can be considered for mitigating a single transition. In the example shown, the risk being explored is loss of information integrity in a spreadsheet. This would be only a part of some system's total risk model. Presumably, the analyst has deemed this high enough in likelihood and consequence to warrant this level of breakdown and analysis.

Barrier analysis is the instantiation and refinement of a risk mitigator's ability to mitigate system specific risks. While the analyst may draw on the Risk Mitigators Matrix for nominal information on barriers, the information may need to be adjusted or supplemented to reflect how the barrier will function in the particular system at hand. Several characteristics of risk mitigators are considered, including how much technology vs. how much rules-and-procedures (rap) are involved, perceived strength, cost to implement, ease of use, outside dependencies. In the example, preventing the user from accidentally altering data by requiring a visual scan is entirely rap, not very strong, and hard to use.

Providing the user a "diff" tool is a stronger technology, easier to apply, and still has some rap component (the user must remember to use it). Providing some sort of automatic overwrite check is stronger yet, has even less of a rap component (the user must still respond appropriately), but may be implemented in such a way as to have a high annoyance factor which may cause the user to ultimately defeat it. All these considerations lead to an estimate of each barrier's ability to reduce the likelihood of transition to the undesired next state.

Threat agents may be active or passive. Active threat agents may have characteristics such as motivation, skills, knowledge, time and other resources. They are willing to incur some amount of cost and risk, in order to gain the perceived value of their target. Passive threats, representing unintentional faults in the system, may have other characteristics. In either case, as a threat unfolds into the system, the agent's characteristics and the system's surety elements may both be altered by the interactions.

The analysis engine combines transition probabilities, threat estimates, barrier estimates, and risk reduction requirements to yield information on remaining risk. In particular, the engine identifies paths in the risk model where risk is still too high. Uncertainty analysis accompanies the calculations, so that the engine can target highly uncertain calculations for refinement. If costs are included for the barriers, then the analysis engine can also produce cost/benefit information.

Iterative refinement is basic to the way in which an analysis should be carried out. The analyst should first work in the context of a high level system risk model, input estimates, run the analysis engine, and examine the results. High risk paths can then be strengthened with additional barriers, or can be broken down into more detail. Highly uncertain paths that are determined to be of sufficiently high consequence call for better estimates, and they may also benefit from refinement. Once the highest level risks have been adequately addressed, the analyst may wish to incorporate additional risks into the model and continue the analysis. The analyst will be able to see the total impact of old and new barriers on all risks.

Tool Support

In order for a rich and comprehensive methodology such as this to be viable, good tool support is a must. A Windows-based toolset, RBASIS (Risk Based Analysis of the Surety of Information Systems), has been developed, consisting of a matrix builder, model builder, analysis engine, and output engine. It has been constructed for flexibility, particularly in the analysis and output engines, so that different computations and formats can be explored.

A Critique of the Methodology

In this section, the methodology is critiqued and compared to other related research.

The Risk Matrices. Others have produced categorizations similar to our "Surety Objectives" and "System Aspects", notably in references [1] and [4]. In [4], Parker introduces a list of "security attributes" consisting of confidentiality, authenticity, integrity, utility, and availability, which he quite reasonably derived by subdividing the traditional security definition of confidentiality, integrity, availability. The differences in his list and our Surety Objectives could be summarized as: we added safety, replaced confidentiality with the broader concept of access control, and did not call out authenticity. While all such lists are arguable, the differences are subtle and certainly not critical for carrying out a risk analysis. Our goal is for the matrices to guide thinking into all relevant areas, so it is not necessary for categories to be totally independent.

The more interesting list is the System Aspects: here, we feel we are uniquely broad. In [1], a perspective is taken which is based on protection of "assets", defined as hardware, software, and information. Then "impacts" to assets are considered, such as destruction, modification, and disclosure, and how such impacts might be mitigated, such as threat reduction, vulnerability reduction, detection and recovery, etc. Parker starts with the assets list, expands software into applications and operating systems, adds users, and renames it "levels of abstraction." [4] We attempted to be even broader, to get away from the static protection-of-assets view, and to introduce system architecture and dynamics. The "Composition" aspect, for instance, includes risks that would be inherent in certain choices of platforms, network, and communications architectures. "State Changes" includes risks that might be present in maintenance procedures and in cases of abnormal system shutdown, etc. The "Interfaces" aspect brings in the context in which the system actually functions.

We hope that this combination of Surety Objectives and System Aspects will encourage the analyst to take a dynamic, whole system, whole lifecycle perspective on risk management. For example, while a narrow view of sabotage might focus on virus protection in an operational system, a whole lifecycle view will encompass protection throughout design, implementation, delivery, and maintenance. And while a narrow view of network security might focus on encrypting communications, a whole system view will explore whether network nodes have compatible security policies and whether they exchange sufficient security information to uphold the policies. And while a narrow view of integrity might address mechanisms within a properly operating database, a dynamic view will also look at shutdown-startup synchronization issues.

The third dimension of our Risk Identification Matrix, Consequences, is kept separate from probabilities throughout the analysis. Those who are interested in computing loss expectancies typically multiply probabilities and consequences. This kind of forced data reduction is not useful for the risk management we hope to accomplish.

In the cells of the Risk Identification Matrix lie Risk Sources. Recall that the intended purpose of the matrix is to guide the analyst in identifying risks, and to initiate the construction of a System Risk Model. The model depicts progressions through various states. Risk sources from the matrix map to states in the model, but the analyst may have to add additional states around what has been mapped in. If hierarchies of risk sources (states leading to other states) can be provided in the matrix, then these can be mapped to subgraphs of the model, giving more of a head start on building the complete model, and the process can be automated to some degree. We believe this is a fairly unique approach to model building. The risk hierarchies constructed to date will be presented at the conference.

The System Risk Model. We believe this risk-state modeling approach is unique in its application to software system risk assessment. The form of the diagram in Figure 3, which we prefer for its visual simplicity, does not convey the interaction complexity that is actually possible in the model. Each barrier can actually influence transitions anywhere in the model, not just between two states, as the figure might imply. This approach is patterned somewhat after the "Influence Diagram" which is gaining in popularity in the probabilistic risk assessment community. [3]

The Analysis Engine. Quantification of probabilities, barrier effectiveness, and threat characteristics is certainly difficult. There is almost no data today, and as such data begins to be collected, it will have a high degree of uncertainty. This is why the analysis engine must accommodate varying degrees of uncertainty and granularity (qualitative - quantitative) in its logic and math. Any math that is applied to probabilities, barriers, and threats is rightfully arguable. We are essentially producing a complex theory of risk, which has no physical basis on which to be validated. This is indeed troublesome. The RBASIS tools are built to be flexible, so that different math and logic can be tried. Even if such a theory of risk can never be validated to a satisfactory degree, we hope that this methodology will be used in a qualitative sense to produce better risk-managed systems.

The Output Engine. The output must be in a form useful for decision making. It is used to improve the system via further risk reduction, to add data or detail where the analysis is weak, and to document risk-based design decisions. These up-front requirements on the output are, to our knowledge, much more challenging than typically found for risk analysis tools. Thus, we felt the tool design warranted a separate output engine. Just as the RBASIS analysis engine is built flexibly to allow for experimentation and improvement, so is the output engine.

Trial Applications

The first trial application is applying the methodology to the RBASIS toolset itself. Consider that we are aiming to provide the software community a tool for risk management. Surely there are risks in trusting such a tool -- risk to the user's mission, and perhaps regulatory or social (embarrassment) risks as well -- should the tool mislead

the user about system risks. Because the tool needs to be a sound and useful product, it was subjected to a risk analysis. This analysis is carried out at a high level to guide the design and development of the toolset. This analysis and others will be presented at the conference.

Summary

The risk assessment methodology presented here is intended to facilitate risk-based design of software systems, and, as well, risk management strategies for the lifetime of the system. It is unique in its broadness, bringing together a wide range of Surety Objectives and System Aspects into a framework where interactions and tradeoffs can be considered. It provides a structured approach to exploring the risk space, documenting risk reduction objectives, exploring alternatives for risk reduction, and documenting design decisions.

Challenges remain in obtaining quantified data for risk analyses and validating mathematics on that data. Meanwhile, the methodology is supported by a toolset which accomodates a spectrum of qualitative-quantitative analyses.

References

1. Proceedings of the 4th International Computer Security Risk Management Model Builders Workshop. August 6-8, 1991. Sponsored by NIST and the University of Maryland.

2. Fletcher, S. K., "The Risk-Based Information System Design Paradigm," Proceedings of the IFIP SEC'94 Conference, May 23-27, 1994, Curacao, NA.

3. Jae, M., and Apostolakis, G. E. "The Use of Influcnce Diagrams for Evaluating Severe Accident Management Strategies," Nuclear Technology, Volume 99, 1992, pp 142-157.

4. Parker, D., "Restating the Foundation of Information Security," Proceedings of the 14th National Computer Security Conference, October, 1991, Washington DC.

5. Stoll, F., "The Need for Decentralization and Privacy in Mobile Communications Networks," Proceedings of the IFIP SEC'94 Conference, May 23-27, 1994, Curacao, NA.

6. Vahtera, P. and H. Salmi, "Security in EDI Between Bank and It's Client," Proceedings of the IFIP SEC'94 Conference, May 23-27, 1994, Curacao, NA.