# An Open Framework for Risk Management[1]

(a paper for the NISSC)


## Abstract

Risk assessment methodologies are ready to enter their third generation. In this next generation, assessment will be based on a "whole system" understanding of the system to be assessed. To realize this vision of risk management, we have begun development of an extensible software tool kit. This tool kit breaks with the traditional approach to assessment by having the analyst spend the majority of the assessment time building an explicit model that documents in a single framework the various facets of the system, such as the system's behavior, structure, and history. Given this explicit model of the system, a computer is able to automatically produce standard assessment products, such as fault trees and event trees. This brings with it a number of advantages relative to current risk management tools. Among these are a greater sense of completeness and correctness in assessment results and the ability to preserve and later employ lessons learned.



Rick Craft (POC)
Surety System Department
Sandia National Labs
505-844-8873 (voice)
505-844-9641(fax)
rlcraft@sandia.gov

Greg Wyss
Risk Assessment and Systems Modeling Dept.
Sandia National Labs
505-844-5893 (voice)
505-844-3321(fax)
gdwyss@sandia.gov

Ruthe Vandewart
Decision Support Systems Architectures Dept.
Sandia National Labs
505-844-7798 (voice)
505-284-3850(fax)
rlvande@sandia.gov

Don Funkhouser
Decision Support Systems Architectures Dept.
Sandia National Labs
505-844-9136 (voice)
505-284-3850 (fax)
drfunkh@sandia.gov



[1] This work was performed under the Laboratory Directed Research and Development Program at Sandia National Laboratories, a multiprogram laboratory operated by Sandia Corporation (a Lockheed Martin Company) for the United States Department of Energy under contract DE-AC04-94AL85000.

# An Open Framework for Risk Management

## Abstract

Risk assessment methodologies are ready to enter their third generation. In this next generation, assessment will be based on a "whole system" understanding of the system to be assessed. To realize this vision of risk management, we have begun development of an extensible software tool kit. This tool kit breaks with the traditional approach to assessment by having the analyst spend the majority of the assessment time building an explicit model that documents in a single framework the various facets of the system, such as the system's behavior, structure, and history. Given this explicit model of the system, a computer is able to automatically produce standard assessment products, such as fault trees and event trees. This brings with it a number of advantages relative to current risk management tools. Among these are a greater sense of completeness and correctness in assessment results and the ability to preserve and later employ lessons learned.

Key Words: Risk Management, Risk Assessment, Risk Analysis, Reliability

## Introduction

If one examines the computer security literature over the last 25 years, the methods and practices employed for information system risk management have changed dramatically. In her paper presented to the 1995 New Security Paradigms Workshop [F95], Dr. Sharon Fletcher asserted that risk management has gone through two generations and needs to enter its third. First generation approaches to risk management grew up in the era of centralized mainframes. These approaches assumed a fixed threat environment and measured a computing facility's security by assessing how well the facility adhered to a prescribed set of safeguards.

As LANs proliferated and computing environments became more distributed, first generation approaches were rendered obsolete. Distributed computing gave adversaries greater access to an organization's computing resources. Second generation thinking – typified by the general risk framework developed in the NIST workshops [N95] – formalized six concepts in risk analysis: (1) assets, (2) vulnerabilities, (3) threats, (4) impacts, (5) likelihoods, and (6) safeguards. Tools based on this approach codify expert knowledge about relationships between these six concepts. To assess a given system, the analyst applies the tool's encapsulated expert knowledge against a catalog of system assets to obtain a list of issues that need to be addressed. Thus, second generation systems began to allow the analyst to tailor assessments on a system-by-system basis.

Third generation approaches, Dr. Fletcher suggests, should take a "whole system" approach to assessment. These methods, and the tools that will embody them, define systems more broadly, and will apply over a system's entire life cycle. Rather than a system being merely a collection of assets, third generation tools will document the system's purpose and behavior, structure, relationship to its environment, and history all in a common framework. Rather than supporting the assessment of operational systems only, Dr. Fletcher asserts that these tools will help system stakeholders "build the right thing, build it well, and protect it appropriately."

To implement Dr. Fletcher's vision of a third generation analysis paradigm requires that we rethink our approach to risk management and the tools that support it. The questions of "What capabilities should such these tools possess?" and "How would these tools be structured?" require significant new thought and planning. Since the Fall of 1996, our organization has sponsored an internal research and development project aimed at answering these questions. The goal of this project is to produce an extensible tool kit for the support of third generation risk management. Members of the research team are drawn from both the information security and the risk and reliability assessment communities, and the framework is intended to be usable for the assessment of many types of complex[2] systems. This paper first summarizes our approach to the third generation analysis framework, and then describes how it is planned to be implemented in a software tool.

## A Framework for Risk Management

The starting point for understanding the third generation approach is to define a framework that captures the various activities that can occur during the risk management life cycle. As this framework is meant to be encompassing, an activity's presence in the framework means that it is used in *some* assessments, but not that it will be used in *every* assessment. Whether or not an activity is used depends on what the goal of the assessment and on the nature of the system being assessed. In presenting this framework, we acknowledge that none of the tasks presented here is new. In fact, many of the tasks are taken from work done for the Commission of the European Communities [G92]. What is different here from what is generally found in the literature is the notion that all of these tasks would be performed in the context of a single system assessment.

The framework that we use is as follows:

- Understand the system
  - Understand Its Behavior
  - Understand Its Physical Structure
  - Understand Its Environment and Spatial Relationships
  - Understand the Role of Timing in the System
  - Understand the History of the System's Components
  - Understand Which System Elements Serve Protective Functions
- Establish Surety[3] Objectives
  - Identify Stakeholders
  - Elicit Surety Objectives
- Understand Component Vulnerabilities
- Characterize Threat Agents
- Assess the System
- Rank Assessment Findings
- Safeguard the System
  - Identify Constraints on Safeguards
  - Evaluate and Rank Candidate Safeguards

---

[2] By "complex", we mean systems composed of diverse sets of technologies – not just information systems, but information systems AND electronics AND pumps and pipes AND … .

[3] This is a term used to mean the combination of safety, security, and reliability.

## Understand the System

Sound assessment is predicated on the analyst understanding the system being assessed. Depending on the nature of the assessment, the analyst may need to understand the system from different points of view. These include knowing what the system does and why, knowing what elements are used to build the system, knowing the nature of the environment in which the system operates, and knowing where each component in the system has been prior to its inclusion in the system being assessed[4].

***Understand Its Behavior*** -- In many systems, the analyst's first understanding of the system is in terms of the functionality that the system delivers. This understanding can be largely independent of knowing specifically *how* this functionality is delivered. This view of a system is particularly useful for identifying potential places to attack or protect in the system[5]. Questions addressed by the analyst in documenting this view of the system include:

- What functions does the system perform, and how are they aggregated?
- What "stuff" flows[6] between blocks of functionality?
- Is the functionality invariant or does it change as a result of various conditions?

***Understand Its Physical Structure*** -- The second view of the system documents the system's physical structure. This is the classic "identify the assets" view prescribed in many first and second generation approaches to risk management. The assets can be computing hardware, software, data items, people, or anything that would qualify as the product of an implementers hand. Once these assets and their interrelationships have been defined, the analyst specifies how these components relate to the functional view. Thus, the analyst must specify which flows and blocks of functionality are embodied in each physical component. Note that this mapping of function to structure can be dynamic. For instance, in some distributed computing environments, a given process can move amongst several computers depending upon service conditions. Questions addressed by the analyst in documenting this view of the system include:

- Of what things is the system composed, and how are they interconnected?
- What are the relationships between these components? Which components exist in peer relationships? In system/subsystem relationships? In " X supports Y" relationships?
- In what physical component(s) is a given block of functionality embodied?
- Is this always the case or can this functionality move from one physical component another?

***Understand Its Environment and Spatial Relationships –*** Every physical system exists in a context. The nature of the environment in which a system operates can significantly impact the system's surety. For this reason, an analyst may need to document the environment in which a system exists. This documentation consists of defining named regions in the system's environment and identifying what things populate these regions. Note that a single component (e.g., a cable) can exist in multiple regions of the environment, and a given region can host multiple components. Physical form and spatial relationships between elements can also be important (e.g., in a communications analysis, "Can this signal be detected at this point in

---

[4] Much of our thinking regarding "understanding the system" has been influenced by the work being done at the Queensland University of Technology. See [C92, A94, K96].

[5] This view is also useful when the system is completely abstract, such as in the early stages of a system's design.

[6] While flows in the system may be information, they may also be material or energy. Our experience has shown that, in complex systems, all three types are usually represented.

space?”). Spatial relationships are particularly important for physical security assessments. Questions typically addressed in this task include:

- Is there a significant partitioning of the environment in which the system operates?
- What are the relationships that exist between these partitions?
- What elements (both inside and outside the system) populate each partition?
- What are the dimensions of each element? How do they relate to each other in space?
- Can the environmental and spatial relationships change over time? If so, under what conditions does this occur?

***Understand the Role of Timing in the System –*** Depending on the nature of the system being assessed, timing issues can play a role in determining the surety of the system. To address this, the analyst documents the normal, minimum, and maximum “propagation delays” associated with the various functions in the system. This information can then be used in analyses to determine if “race conditions” can drive the system toward undesirable outcomes.

***Understand the History of the System’s Components –*** In some applications, the analyst needs to know what has happened to a component prior to its inclusion in the system being assessed because actions that occurred in one stage of a component’s life cycle can adversely affect its performance in subsequent stages. To account for these issues, the analyst can document and analyze a component’s life cycle. Questions that the analyst asks in this task may include:

- Could a given component in the system being assessed have been subverted prior to the component’s inclusion in this system?
- If so, at what points in the component’s life cycle could this have occurred?
- What is the nature of the system at each of these points (this can cause the analyst to ask for this new system the entire set of questions asked to understand the main system)?

***Understand Which System Elements Serve Protective Functions –*** In existing systems, some system elements may exist solely to safeguard the system. If the analyst wishes to compare the effectiveness of existing safeguards with proposed changes, he may wish to remove the existing safeguards from the model and analyze a “bare” system as a “baseline” case. During the safeguards assessment, the existing safeguards can then be reintroduced as one of a number of safeguard suites to be evaluated.

### ***Establish Surety Objectives***

In addition to understanding the system being assessed, the analyst needs to identify the system surety objectives. These come in two flavors: those things that must be prevented (e.g., the unauthorized disclosure of a given piece of information) and those things that must be assured (e.g., a specified level of system availability). In defining these objectives, the analyst may first need to identify stakeholders, elicit objectives from each stakeholder, and then combine and prioritize the list of objectives.

***Identify Stakeholders –*** Since a single system can impact multiple individuals or organizations, an analyst may need to collect surety objectives from more than one stakeholder. Thus, the first step in establishing system surety objectives is to identify stakeholders. Note that the definition of a stakeholder can vary from system to system. In general, stakeholders can include system users, persons affected by the system, people who monitor or regulate its use, or entities that pay for the system's development or operation.

***Elicit Surety Objectives –*** For each element of the system model, the analyst asks each stakeholder what objectives exist for that element. While some objectives will be expressed in terms of the physical structure of the system or its environment, most (but not all) can be traced back to the functional view of the system. The objectives then need to be compiled and prioritized. Conflicting objectives are brought back to the stakeholders for discussion and resolution. The objectives can be prioritized on the basis of either explicit cost metrics or simple negotiation among the users. Questions that the analyst asks in this task may include:
- For this element or flow, are there any things that we must prevent or assure? If so, why?
- Which of these objectives are mandatory, and which are optional?
- Do any of the collected objectives conflict? If so, how will conflicts be resolved?
- What is the relative importance of the various objectives? Why?

### ***Understand Component Vulnerabilities***
Just as the analyst must understand a system as it is supposed to exist, the analyst must also understand how its individual elements can fail or be subverted. As the system can be understood from different points of view (functional, physical, etc.), this task inherently spans all views. In looking at the functional view, the analyst asks questions regarding the characteristics of abnormal flows or the effects of corrupting given chunks of functionality. In the physical view, the analyst catalogs weaknesses in the components and considers what other aspects of the physical component bear on the system's behavior. In the environmental view, the analyst considers how changes in the structure or relationships between environmental elements can affect their behavior. Questions that the analyst asks in this task include:
- How can a flow be perturbed? How can this perturbation be caused?
- How can relationships (e.g., peer, system/subsystem) between system elements be altered?
- What is the effect of each perturbation or change on the system and its associated elements?
- Can an element exist in a fault state? How does the fault state affect the element's behavior?
- What immediate influences could cause the element to enter a fault state?
- Can a system element respond to any other flows that are not part of the normal system model but that cause the element to fail or be subverted?

### ***Characterize Threat Agents***
Once the model of the system has been elaborated to include the fault behaviors of each system element, the analyst has a basis for identifying and characterizing potential threat agents. In identifying potential threat agents, the analyst considers each flow in the system and asks, "What agent is capable of producing or influencing this flow?". Both "passive" and "active" threats must be considered. Passive threats often have their genesis in the environment (e.g., fire, flood, etc.), while active threats exhibit intelligence. While these are typically humans, they can also be non-living entities, such as software agents. In characterizing threat agents, the analyst always documents the agents' capabilities. For active threats, the analyst also documents factors such as motivation and risk aversion. Questions asked by the analyst in this task include:
- What agent is capable of producing or influencing a flow or system element?
- Which elements (inside or outside the system model) could be this agent?

- What are capabilities of this threat agent?[7]
- If this is an active threat agent, what are its characteristics (objectives, risk aversion, knowledge of the system, etc.)?
- If a threat agent has any capabilities that are not currently modeled in the system description, could these capabilities be significant to the functioning of the system?

### ***Assess the System***

Given that the analyst understands the system, knows how the components can fail or be subverted, and has identified system surety objectives, the analyst can assess the surety of the system as a whole. Assessment approaches available to the analyst come in four varieties: deductive logic techniques, inductive logic techniques, heuristic searching, and simulation.

***Deductive Logic Techniques –*** In deductive logic techniques, the analyst selects a given surety objective and then chains backwards (against system flows) starting from the point in the system where the objective manifests itself. Here the analyst seeks to determine what event sequence(s) could cause the objective to be subverted. Fault tree analysis is typical example of this approach.

***Inductive Logic Techniques*** – In inductive logic techniques, the analyst specifies one or more initiating events and then chains forward (in the direction of the system flows) in order to determine the universe of possible system outcomes to which the events can lead. Event tree analysis and Failure Modes and Effects Analysis are typical examples of this approach.

***Heuristic Searching –*** In heuristic searching, the analyst defines a threat agent that is to attack the system. The agent has an initial set of capabilities, some degree of access to the system, and one or more goals to reach. Given this, the agent begins interacting with the system model in an attempt to gain additional capabilities and to increase its level of system access until its goals are reached.

***Simulation –*** Simulation is used to give the analyst more insight into the system than might be gained through static models of the system. In simulation, the analyst sets the initial state of the various system elements and then "starts" the system model. The analyst can then alter specified parameters in the model and observe the associated outcomes.

### ***Rank Assessment Findings***

The results of an assessment generally consist of  a set of "incident scenarios". In the classic probabilistic approach to ranking, each scenario is associated with a consequence and likelihood of occurrence. The consequence measures ("costs") are identified by the stakeholders (often through an iterative process), and are often based upon actual dollar loss values, although other metrics, such as lives lost or ecological damage, may be more appropriate for some systems. The likelihood of occurrence helps provide a context for the consequence measure. This measure may be either qualitative or quantitative, conditional or unconditional. It may be an actual frequency of occurrence or simply the relative difficulty that an attacker would have in executing the scenario (determined based on relationships between the threat agents and the elements that they

---

[7] This is done to identify those capabilities that an agent may possess but that are not documented in the set of capabilities defined during the cataloging of threat agents

attack). The consequence and likelihood values are then used to determine the risk[8] each event sequence represents to the system. Alternatively, the analyst can choose other approaches to establishing an ordering of the assessment finding. Different decision theoretic approaches, combinations of approaches, or even "expert judgement" can be used in each situation as appropriate. Questions associated with this task may include:

- What are the costs associated with failure to meet this objective?
- How important is one cost relative to other costs in the system?
- In relative terms, which outcome is most at risk?
- What system elements or flows contribute most to the aggregate system risk?
- Which system element is found in the greatest number of undesirable outcomes?

***Safeguard the System***

Once a ranked list of system problems exists, the analyst can begin the process of selecting safeguards. This process consists of identifying any constraints that might be levied on the safeguards process, of identifying candidate safeguards and evaluating their effectiveness, and then ranking safeguard results to identify the best options.

***Identify Constraints on Safeguards –*** Typically, constraints are levied that specify some aspect of the safeguards that can be applied to a system. These constraints can include limitations on the amount of money to be spent on safeguards, constraints on reduction in system performance introduced by safeguards, usability requirements, power and space requirements, and legal requirements. The process of identifying these constraints is much like the process of setting surety objectives: stakeholders are identified, constraints are solicited, and a final prioritized list is produced through negotiation among the stakeholders. Questions addressed in this task are:

- Who is impacted by the use of a safeguard at a given point in the system? How?
- What resources (money, bandwidth, CPU cycles, space, power, manpower, etc.) can be used in a system element to design, purchase, operate and maintain elements that support surety?
- What legal, regulatory or political constraints apply to the safeguards process?
- How firm are each of these constraints, and how will conflicts between them be resolved?
- What is the relative importance of the various constraints? Why?

***Evaluate and Rank Candidate Safeguards –*** Given the ranked list of problems to be solved and the ranked list of constraints, the analyst identifies candidate suites of safeguards to be applied to the system. If the system being assessed already had safeguards that were stripped out before assessment, then these safeguards are considered to be one candidate safeguard suite and are evaluated like any other suite. For each of the suites applied to the system, the analyst runs the same system assessments that were performed on the bare system in order to create a consistent basis for comparison. The ranking techniques applied to the bare system are then applied to the safeguarded systems. The relative mitigation effects are then assessed by the analyst, and a specific suite selected for implementation. Questions addressed in this task include:

- Which safeguards will address the ranked system problems within the identified constraints?

---

[8] Risk is a function of likelihood and consequence. Originally the simple product of likelihood and consequence, it is now most frequently viewed as a more general (possibly nonlinear) function that more accurately expresses the stakeholders' tangible and intangible values related to the system under analysis. It can be thought of as being similar to the utility function in an optimality analysis.

- How should these individual safeguards be combined into suites to achieve the best mitigation effect subject to the constraints?
- What are the relative strengths and weaknesses of each suite?
- Which suite gives the best overall reduction in system risk?
- Which suite gives the most balanced approach to security?

## An Open Tool Kit

To our knowledge, there is no tool available today that supports the full set of tasks specified in the framework. In particular, we have not found any that adopt a "whole system" approach to understanding the system[9] nor have we found any that allow us to deal with complex systems. For this reason, we began our research and development project aimed at producing a tool that would deliver the capabilities that we have described.

We discovered in the course of our research that concepts found in the object-oriented analysis arena go a long way to meeting our needs. In particular, "objects" can be used to document a system's behavior, its structure, its environment, and how these different views of the system relate to one another. With proper structuring, objects can be used to document component life cycles, to capture the information needed for timing analysis, and to model the fault mechanisms and vulnerabilities found in system components. In fact, with the exception of those aspects of a model related to a system element's physical dimensions, orientation, and location, we have found the language elements found in object-oriented analysis to constitute a suitable language for development of our risk management tool.

In the developing this tool, we have adopted two main philosophies. First, the tool should permit an analyst to use the right methods at the right time. Just as a hammer is not always the right tool for the job at hand, a specific assessment method will never be universally applicable. This has lead us to implement a "tool kit" instead of a monolithic tool. Our intent is to permit tools in the tool kit will be used in mix-and-match fashion and to minimize the effort required to add new tools in the future. Second, we believe that the folks at QUT are correct when they assert that "it is more productive to perform a relatively simple analysis on a comprehensive [system] model than to undertake sophisticated analysis of a simplistic one" [K96]. Because of this, the starting point of an assessment using our tool kit is the development of an explicit system model.

With respect to this second point, we have found that if the analyst takes the time to explicitly document what he understands about the system being assessed, several benefits are realized. First, the requirement to "hand tool" each assessment disappears or is minimized. For example, in fault tree analysis, the analyst begins by defining a top-level event and then works backwards through the cause-and-effect links to determine what chain(s) of events could result in the top-level event. In current manual assessment methods, each time the analyst defines another top-level event, he must again go through the process of tracing the causal chains back to their initiating events. In contrast to this, if an explicit system model has been developed, the generation of the fault trees can be done automatically by a computer. All the analyst needs to do is to identify the top-level events of interest and the computer does the rest. In fact, the computer can use the same model base with a range of assessment techniques. We are currently convinced

[9] The "Risk Data Repository" being developed at Queensland comes closest to what we are looking for.

that given a single explicit model of a system, we can automatically develop the assessment models for a range of techniques including fault tree analysis, event tree analyses, HAZOP analysis, failure modes and effects analyses, and vital areas analysis.

The second benefit to using an explicit system model is that many of the features that people have wanted in assessment tools are finally realizable. In particular:

- ***assessment results become traceable*** **–** a person can see how the analyst got to a particular set of conclusions
- ***there is a sense of completeness to the assessment*** **–** the use of an explicit model makes it possible to see what has been investigated and what has not
- ***it is possible to assess the correctness of the assessment*** **–** since the model is explicit, it is easier for system stakeholders to determine whether or not the models reflect reality

In traditional assessment approaches most of what the analyst understands about the system exists only in the analyst's head; consequently, assessment details, like how the analyst linked attacks to outcomes, are often hidden. Similarly, if issues seem to be missing from the assessment results, it is not clear whether the analyst understood these issues and considered them inconsequential or whether the analyst was simply oblivious to the critical facts underlying these issues.

The third benefit is that management of assessment information becomes an easier task for the analyst. Even in small systems, the number of facts that need to be gathered and retained can be overwhelming. While no one fact may be difficult to understand, the sheer volume of facts can often make understanding the "big picture" difficult. To address this problem, object-oriented analysis approaches permit a system to be modeled in hierarchical fashion. When needed, the analyst can view a component as nothing more than a block that interacts with other blocks in a system. How the component operates internally is hidden when not relevant to the analysis being performed at the moment. At the same time, when the details of how the component works need to be understood, the analyst can "open" the component to investigate these details and, if appropriate, ignore the details of how the system outside the component operates.

Fourth, the use of an explicit object-oriented model organized makes it possible to preserve "lessons learned". In our approach to modeling, the system's functionality and its structure are modeled as separate objects. For instance, a computer responsible for supporting a portion of a distributed application would be modeled as two collections of components: those that document the behavior of that portion of the application that resides on the computer and those that model the computer. The reason for this is that the vulnerabilities that exist in the computer and that put the application at risk are really properties inherent to the computer and not the application that the computer happens to support. By separating an application's behavior from the components used to deliver that behavior, we can encapsulate what we know about component vulnerabilities within the components themselves and then reuse this knowledge in analyses of other systems built with the same components. For example, if Windows NT is running on a computer that controls a manufacturing process, the analyst may discover that it is possible to corrupt the process by attacking the interprocess communications (IPCs) in a specific way. The effect in the system is to corrupt some portion of the manufacturing process, with the result that products are destroyed and money lost. The analyst may later assess another system that uses Windows NT and the same types of IPC. When the analyst adds the Windows NT component to the model of this new system, it brings with it knowledge of the IPC attack gained in the previous system assessment. Any assessment that then trace their way to cooperating processes on the NT platform will end up automatically linking in the IPC attack thread.

In developing a system model, the analyst defines a network of cooperating objects. An object in this network can represent a block of functionality delivered by the system, a component from which the system is composed, or an element in the system's environment (such as a building that contains system objects or a threat agent that attacks the objects). Within a view of the system (e.g., the system's functions or its structure) and across views, objects communicate with each other. The communications within a view represent real flows in the system. Communications between views tend to represent "supports" relationships but can represent other things. For instance, a connection between two objects in the physical view of the system indicates that "stuff" flows between these objects. On the other hand, the connection between a functional object and a physical object, like a computer, indicates that the physical object supports that functionality. If the computer is lost, then so is the functionality that it supports. Objects in a system model can also be hierarchically decomposed into constituent objects. As discussed above, this permits an analyst to hide details about an object within the object when not needed for whatever problem is currently being considered. It also makes it possible for the analyst to conduct an assessment without having to have all of the details about the system before beginning to derive results. By modeling a system in hierarchical fashion, the analyst can create a top-level model of the system and assess it to get an initial feel for where problems might exist. The objects in the system that seem to be problematic can then be decomposed and developed more fully as the analyst explores the details of these components.

In our approach to modeling, when an object receives something from another object, it responds in some fashion. This response can include ignoring the communication, outputting its own communications, and/ or changing its internal state. To capture this knowledge about an object, we use state charts, data flow diagrams, attributes, attribute values, and decision tables. State charts are used to document the fact that the way that the an object responds to communications can vary with time. The way in which an object acts at a specific point in time is documented with data flow diagrams. Decision tables are used to document the behavior of processes in the data flow diagrams. Attributes and their associated values are used to describe the communications that flow between objects and to document the internal state of the object.

Given this type of system model, we have found that we can use the same language constructs to specify surety objects, to describe system vulnerabilities, to model threat agents, and to create the building blocks of assessment structures, such as events in a fault tree. It is this fact that permits us to create the tool kit in such a way that products of one tool can be manipulated by other tools.

By the time this paper is presented, we expect to have completed a first version of the tool kit. In this initial incarnation, we intend to use the Rose CASE tool from Rational Software to support modeling of the system's functional, physical structure, and environmental views. Custom applets will be used to specify system objectives and to control assessments. A risk analysis engine called ARRAMIS will be used to analyze fault trees and event trees pulled from the models developed in Rose. The process steps that address safeguarding the system will not be addressed due to limits on time and manpower. In this version of the tool kit, the model base created by Rose will serve as the tool kit's model repository. The custom applets and ARRAMIS will read from and write to this model base.  In subsequent versions of the tool, we intend to create a separate, stand-alone repository.  We have already acquired the Versant object-oriented database for this purpose.

## Conclusion

Risk assessment methodologies are ready to enter their third generation.  By taking a broader view of what constitutes a system, these methodologies increase the workload levied on the analyst; therefore, automation support will be a necessity.  We believe that techniques developed in object-oriented analysis world and embodied in CASE tools can be used effectively to develop the explicit systems models from which standard assessment mechanisms can be automatically derived.  One of the primary benefits of this approach will be the ability to deliver higher quality assessments than is possible now.  This approach also makes it possible to retain (and even exchange) expert knowledge regarding component vulnerabilities and failure mechanisms.  A tool intended to embody this approach to assessment is in development and an initial version of it should be available by the end of the current fiscal year.

## Bibliography

[A94] Anderson, A., Longley, D., and Kwok L. F., "Security Modeling for Organisations," Proceedings of the 1994 ACM Conference on Computers and Communications Security, November 1994, Fairfax, Va.

[C92] Caelli, W., Longley, D., and Tickle, A., "A Methodology for Describing Information and Physical Security Architectures," Internal Report of the Information Security Research Center of the Queensland University of Technology, April 7, 1992

[F95] Fletcher, S., Jansma, R., Lim, J., Halbgewachs, R. Murphy, M., Wyss, G., "Software System Risk Management and Assurance," Proceedings of the 1995 New Security Paradigms Workshop, August 22-25, 1995, San Diego, CA.

[G92] Glover, I., "Commission of the European Communities Security Investigations Projects: Project S2014 – Risk Analysis", Proceedings of the 1992 NIST Risk Model Builders Workshop, March 30 - April 1, 1992, Ottawa, Canada.

[K96] Kwok, L. F. and Longley, D., "A Security Officer's Workbench," Computers and Security, Vol. 15, No. 8, pp. 695-705, 1996

[N92] NIST, Proceedings of the 1992 NIST Risk Model Builders Workshop, March 30 - April 1, 1992, Ottawa, Canada.