USE OF THE ZACHMAN ARCHITECTURE FOR SECURITY ENGINEERING

Ronda R. Henning
Harris Corporation
Information Systems Division
Mail Stop W2/7756
P.O. Box 98000
Melbourne, FL 32904
407-984-6009

1.0 Introduction

A system security policy is often perceived as a set of mandatory requirements levied upon the system by an organizational directive or Information System Security Officer (ISSO). To the user, these security requirements may bear little resemblance to his actual working system security policy, which controls data modification and user privileges. In the course of reengineering business processes and information systems, the system modeling activities provide a unique opportunity: This paper presents a methodology for security policy definition using the Zachman information systems architecture as a tool. The system security policy can be extracted from the Zachman framework, providing a technique for reconciling the security policy as defined by directive with the user's working system security requirements.

2.0 Security Policy Derivation -- Today

In the current generation of system specifications, the security policy requirements are often summarized as the requirements for operation in a given secure mode of operation as specified in an organization's security guidance. From the accreditor's or designer's perspective, a system is defined as running in a given mode of operation, and at a given classification level (or range of levels). A system security policy may be divided into the individual sub-policies, for example: identification and authentication, auditing, access control, and network access. Additional specification detail may be presented, such as: the only auditable events are "login" and "logout;" or the system must protect itself from malicious code or virus infestation. Figure 1 illustrates a possible decomposition of a system security policy into subpolicies.

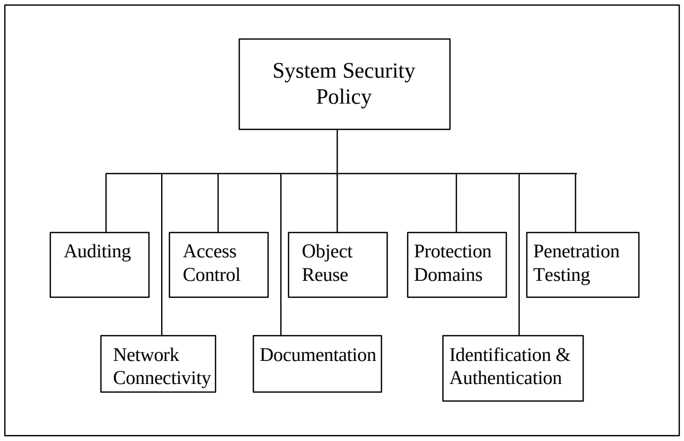


Figure 1. A decomposition of a system security policy.

These requirements may be levied by regulation, not necessarily operational system usage. For example, a real-time command and control system may not require individual user login and logout, citing operational necessity. A mission simulator that provides scenarios for multiple mission planners may not require the flight crews to login/logout. The words “that’s not how my system works” usually follow the realization that organizational policy requirements must be implemented in the re-engineered system. The user does not identify with the requirements, and perceives security as an obstacle to his mission..

2.1 The Hidden Security Policy

Beyond the explicit security section of a system specification, there may be an implicit security policy. This policy is often couched in phrases such as

- “ensure operators can only perform operations X, Y, and Z,”
- “only mission managers can modify plans,”
- “there are only two roles in the system: operator and user.” and
- “users cannot change system parameters”.

This implicit security policy must be elicited by gathering these phrases from the specification and concept of operations documents. Unfortunately, the end user of the system does not always recognize that the implicit security requirements are security requirements. When an architecture uses Commercial-Off-The-Shelf (COTS) security mechanisms to implement implicit security requirements, the customer may intervene. For example, an access control requirement, implemented through the use of operating system access control lists, a standard COTS mechanism, can be met with a customer comment such as: “That can’t be a security requirement. Security didn’t specify it. The old system has some software written to check if the user is allowed to modify that file.” This approach results in sub-optimal solutions with no inherent system flexibility. In such a scenario, the working security requirements can only be solicited if the data flows are defined and analyzed, placed in the context of the updated system requirements, and then designed into the new system. This approach provides the functionality required by the user, with security mechanisms that can be maintained throughout the system lifecylce. System certification time is also decreased, because it is much easier for a Designated Approval Authority (DAA) to inspect the configuration of access controls within a system as opposed to code inspection of new security mechanisms.

2.2 Security Requirements Synthesis

When the implicit security requirements are coupled with the directive-based security requirements, a true baseline of security requirements for a system emerges. At this point the secure systems engineer applies risk management techniques to determine the relative criticality of the data. This information helps define what protection mechanisms to apply in any given system architecture. For example, if a system processes five percent of the data at the classification TOP SECRET, and the remainder of the data is UNCLASSIFIED; it may be much more cost effective to build a subset of the system to address the TOP SECRET data segregation requirement. In conjunction with the customer, the security architect for any system must derive:

- which data elements to protect,
- how much protection this data requires,
- how this data may be modified, and
- how this data is communicated to other systems.

The captured information is discussed in the Security Accreditation Working Group, and documented in the Security Certification and Accreditation Report. It is used by the DAA to define the security characteristics of the system, and to determine if appropriate safeguards were applied. From the DAA’s perspective, the system architecture must provide protection consistent with the data contained in the system.

Presentation of the information in a cohesive format is the responsibility of the security engineer. Security organizations speak in terms of subjects and objects, with Mandatory Access Control (MAC), Discretionary Access Control (DAC), Identification and Authentication (I&A), and Object Reuse policies, using terminology that is unfamiliar to the majority of system customers. This can make it difficult to reconcile requirements as expressed by the user in terms that are understandable to both the customer and the certification organization. To facilitate this task, the Zachman Model of Information Systems Architecture can be used. The Zachman Model is normally applied to general purpose information modeling tasks. With some forethought, it can readily be adapted to incorporate security policy modeling as a part of traditional information modeling activities.

3.0 The Zachman Framework

The Zachman Framework for Information Systems Architecture (ISA),[1] defined in 1987, is a logical construct to define and control the interfaces and integration of all components of a system. The framework of the Zachman model enables systematic capture of system specific information from the various perspectives with respect to a system architecture. Figure 2 illustrates the 30-cell Zachman model, tailored to support an information systems re-engineering application. In this customization of the model, the system developers have an existing operational system in place. The model is applied to capture the security policy of the existing system to ensure the actual user requirements are understood prior to system re-development. When this framework is complete, the explicit, directive based security requirements can be applied and overlayed into the framework, reconciling the implicit, working model and the directive based model for the system's security requirements.

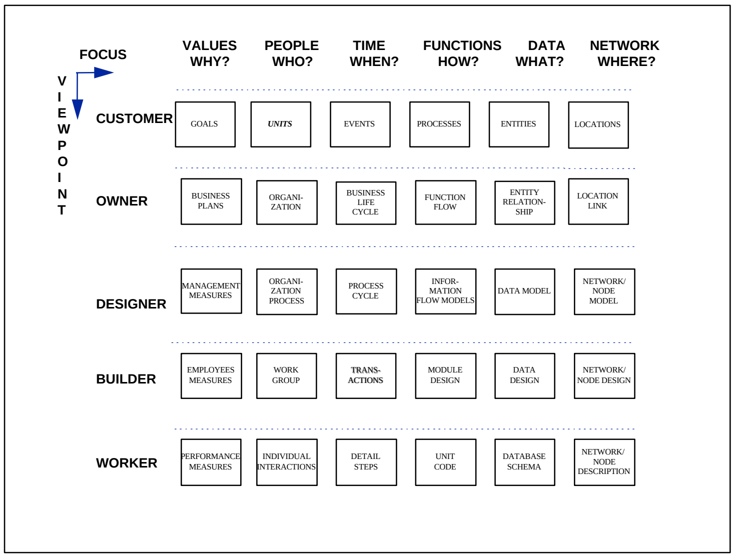


Figure 2. The Zachman Framework for Information System Architecture.

The Zachman framework has two very distinctive features that make it ideal for information modeling. The framework may be applied at any level of abstraction in the system development process, from a global enterprise, to a system, subsystem, or major module level. The framework also gives the modeler latitude in that any data representation technique can be used to model the inner workings of each cell. For example, entity relationship diagrams, IDEF (Integrated DEFinition language, or ICAM (Integrated Computer-Aided Manufacturing) DEFinition Language)[2] models, and conceptual graphs are all equally valid representations of the information contained within a given cell.

---

[1] The Zachman Model was initially described in "A Framework for Information Systems Architecture," IBM Systems Journal, Vol. 26, No. 3, 1987, pp. 276-292.
[2] The IDEF model is described in various publications, including: "IDEF Family of Methods for Concurrent Engineering and Business Re-engineering Applications." Richard J. Mayver Ph.D., Michael K. Painter, and Paula S. deWitte, Ph.D., Knowledge Based Systems Inc., 1993, and "The IDEF

As one changes perspective from the customer level down to the worker level, more detail is provided, and less large scale perspective from the upper cells is visible. The system model becomes more implementation specific. However, the requirements traceability between layers can be maintained through backward references to upper layers of cells. This traceability is critical in security requirements engineering, where tracing a global access control requirement may translate into explicit setting of access controls on specific files or directories within an operating system.

The framework provides a taxonomy "that helps us understand the perspectives of various players in the development of an information system and the descriptions of the system that can be produced during its creation."[3] The model is frequently used as a framework during information systems re-engineering activities to support the solicitation, identification and mapping of the following information associated with an information system's:

- goals, objectives and environment,
- customers served,
- time constraints,
- functional description,
- information architecture, and
- supporting infrastructure.

Application of any model implies a set of rigor and structure. For the Zachman Model of Information Architecture, the basic structural framework rules are:

- The columns in the framework have no order, which would create a bias towards one perspective of the system over other perspectives.
- Each column is based on a simple, basic modeling technique. The columns provide answers to the basic "who, what, when, where, why, and how" questions.
- Columns are unique, that is, their contents are not repeated, which preserves the ability to define a categorization scheme for the model.
- Rows represent a distinct, unique perspective of one of the models (i.e., scope, enterprise, system, technology, component, or working system).
- Each cell is, in itself, unique. So the resulting metamodel is, in itself unique.
- The composite, or integration of all cell models in a single row constitutes a complete model from the perspective of that row.
- The logic is recursive, allowing increasingly more detailed models to be developed.[4]

The resulting information system architecture provides a unique model, where, at any given row level, an integrated perspective of the system can be produced answering "who, what, when, where, why, and how." The framework allows ownership of activities and data to be established, and traced throughout the system development process.

In short, the Zachman Information Systems Architecture can provide a consolidated view of a system, to whatever level of detail a modeler chooses.

3.1 Application of the Zachman Model

Within Harris Corporation's Information Systems Division, an Information Systems Reengineering Action Team was tasked with the definition of a corporate information systems reengineering methodology. The methodology created is based on the Zachman Model, and is used to define the present system, the desired system, and a transition strategy to bridge the user's expectations between the two system models. In the absence of a commercial off the shelf solution, Harris developed a middleware application that automates the support of the Information Systems Reengineering

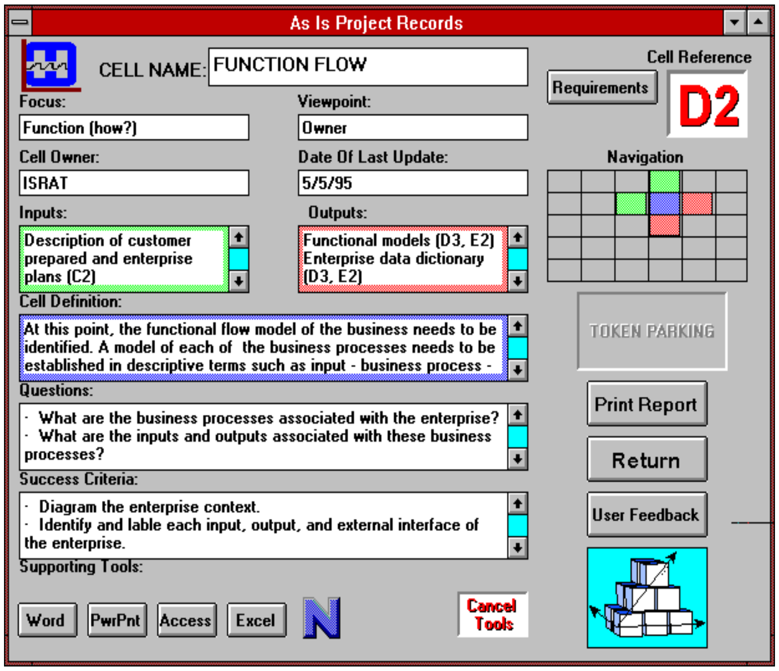
---

Framework Version 1.2," publication of IDEF Users Group Working Group 1 (Frameworks), May 22,1990.

[3] Bruce, Thomas A., "Simplicity and Complexity in the Zachman Framework," *Data Base Newsletter*, May/June 1992, p. 3.

[4] Sowa, J.F. and Zachman, J.A., "Extending and Formalizing the Framework for Information Systems Architecture," *IBM Systems Journal*, Vol. 31, No. 3, 1992.

Methodology. The middleware application provides built-in solicitation for the development of Zachman cell contents. It also supports requirements management by enabling the mapping of the requirements to the same frame of reference, resulting in a requirements repository reflecting the current system and its evolving replacement. The tool does not replace sound engineering discipline, but facilitates requirements capture and interface definition activities. Figure 3 illustrates the top level menu of the Information Systems Reengineering Task Management Tool.



CELL NAME: FUNCTION FLOW

Focus: Function (how?)

Viewpoint: Owner

Cell Owner: ISRAT

Date Of Last Update: 5/5/95

Inputs: Description of customer prepared and enterprise plans (C2)

Outputs: Functional models (D3, E2) Enterprise data dictionary (D3, E2)

Cell Definition: At this point, the functional flow model of the business needs to be identified. A model of each of the business processes needs to be established in descriptive terms such as input - business process -

Questions:
· What are the business processes associated with the enterprise?
· What are the inputs and outputs associated with these business processes?

Success Criteria:
· Diagram the enterprise context.
· Identify and lable each input, output, and external interface of the enterprise.

Supporting Tools:

Figure 3. Information System Requirements Modeling Tool Screen.

In the course of using the tool for generic requirements management, it became evident that applying the Zachman Model to security engineering would be a relatively uncomplicated application. With some minor rework, the model could be readily adapted as a security policy modeling tool, providing a framework for the reconciliation of the implicit and explicit security requirements associated with a system architecture. It could also provide a useful tool to the system security certification team as a requirements traceability matrix. Applied in this manner, the Model traces the top level system requirements specification down to the actual implementation mechanism.

4.0 Security Modeling Integration with the Zachman Model

The Zachman Model cell organization is structured into five levels, or rows, representing increasingly detailed perspectives on the system in question, as defined in the following table.

Table 1. Zachman Model Cell Organization from a Layered Perspective.

Layer	Perspective	Description
1	Customer	Defines a clear and coordinated boundary (domain) of the system for the purposes of identifying people, subsystems, and needs impacted by the system.
2	Owner	Captures the business and organizational relationships, and their external interfaces. Documents requirement sources, including those derived from legacy systems.
3	Designer	Defines functional capabilities of the system and establishes and

		documents the architectural foundation for system design and development.
4	Builder	Establishes and documents the architectural design. Provides basis for system measurement.
5	Worker	Provides detailed description of design and methodology for monitoring and correcting system performance.

For security policy modeling purposes, the first three levels of the perspective hierarchy (customer, owner, and designer) are extremely useful. They provide the consumer perspective of the system's end user, the perspective of the system "owner" or contracting entity, and the perspective of the designer, or systems engineer. In other words, the "as built" and used in daily operation perspective, the "as desired" operation perspective, and "as actually specified" perspective.

One of the more common modeling methods that can be used to define cell content is the IDEF language. The IDEF model, layer 0 (IDEF0) model provides a representation of the inputs, outputs, controls, and mechanisms associated with a given cell. An IDEF0 model of the inputs, outputs, and process constraints associated with each cell can generate additional security relevant information. Figure 4 illustrates the generic IDEF0 model using an external perspective to the cell itself.

Without any additional information, the external prospective provides the data flow through the system, the command media flow down from upper levels, and the mechanisms with associated system performance constraints.

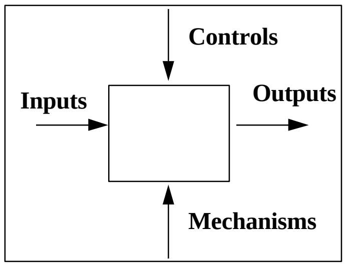


Figure 4. Generic IDEF0 Model- External Perspective.

4.1 Security Derivation

When the top three layers of the Zachman model are applied to a system, without any additional security information, the security engineer can readily obtain:

- the system functions,
- the system information flows,
- the network connectivity of a distributed system,
- the data model,
- the data "owners, modifiers, and users," and
- the responsibilities of organizational entities associated with the system.

If some additional security relevant information is appended to the IDEF0 model constructs, the external perspective illustrated in Figure 5 results, and the model construct becomes more useful for security policy modeling. Annotation of the IDEF model with this minimal additional information provides the security engineer a more robust picture of the potential security problems associated with a system. For example, the customer can claim the system does not in any way, shape or form connect to a system at a higher classification level. If a particular input can be identified as coming from a particular source system, the classification level associated with the source system can be verified. Similarly, a list of user roles or access control rules such as "only users at location X" can be assimilated into the system access control policy. Determination of possible information downgrade procedures can be determined by examining inputs, outputs, and mechanisms for classification.

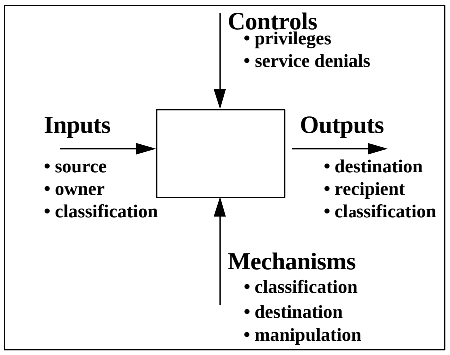


Figure 5. Annotated IDEF0 Model with Security Attributes.

The objective of this exercise is to find possible inconsistencies between the system as described, as specified, and as mandated in governing policy documents as early as possible in the system specification and design process. The goal is to have an accurate representation of the system consistent with applicable security policy and doctrine in effect.

4.2 Applying the lower model layers

While security policy modeling itself is not as concerned with the builder and worker levels of the model, these levels have great value as a security accreditation aid. Through the designer perspective level, a complete set of system security requirements and a security policy specification is defined. Continued application of the model at the builder and worker layers provides requirements traceability down to the level of security mechanism implementation. Table 2 illustrates this correspondence.

Table 2. Correspondence of Upper Layers to Lower Layers.

Level	Perspective	Description
1	Customer	Tasking is received in the system from “System Y.” Crew Chiefs read the tasking and come up with how to fulfill it.
2	Owner	Mission plan is developed by “Crew Chiefs” only.
3	Designer	Mission plans are kept in individual text files. Only “privileged users” (i.e., Crew Chiefs) can modify the mission plans.
4	Builder	Use system discretionary access control to define modification privileges to the file.
5	Worker	Access Control List entry shows “write access”; audit trail record written for all file access attempts.

By providing visibility and traceability across the security requirements, it may be possible to more readily develop system security test plans and ensure comprehensive coverage of the requirements.

4.3 Near term Additions

Security requirements can be levied upon system architectures from directives and policy guidance documents such as Director Central Intelligence Directive 1/16 (DCID 1/16) and the individual service’s security directives. These guidance documents superimpose a set of static requirements upon a system, defining the requirements for a given mode of operation. For example, a system high mode of operations requires individual user identification and authentication. It would be most useful if such static requirements could be incorporated into the basic model template of thirty cells and loaded with the initial Zachman model definition. This would avoid duplication of effort, and eliminate the possibility of "overlooking" a requirement. One template for each mode of operation and each policy directive would be required. Current activity addresses the decomposition of the mode of operation requirements into appropriate cells in the architecture. Then a menu selection of mode of operation will be possible in conjunction with initial population of the hierarchy.

The IDEF0 template fields prompting for input, output, control, and mechanisms could be modified to address the security annotations discussed above. Again, this activity is designed to integrate the security process with the fundamental information and process modeling activities associated with the system. The goal is to make security an integral part of the system design and development activities as painlessly as possible, with minimal impact on the customer and the engineering staff.

Use of the model output as accreditation evidence has not been attempted. Its acceptance as accreditation evidence would be contingent on the diligence of model maintenance. The initial modeling could be translated into a system security policy document in the traditional sense. Doing the model during requirements specification, and not maintaining correspondence between the various model perspectives during system development would make it difficult to submit the model itself as credible as accreditation evidence. If the model is maintained, and traceability among the cells can be demonstrated to the satisfaction of the Designated Approval Authority, the model should be acceptable as part of a system accreditation package.

5.0 Problems in Use

Users of the Zachman modeling methodology have previously discussed the importance of maintaining a consistent perspective on the system across the model[5]. It is easy to become so intent on modeling a given cell that great detail is applied, and other cells may be ignored or minimally addressed. In this case, one of two things has happened: either the customer has provided great detail in his description of one part of the system, or has provided minimal data about the minimally addressed cells.

The recursive nature of the model makes it possible to define complete iterations of the model at varying levels of complexity. In this scenario, one could start with a top layer model of the Air Force, with subsequent layers for major command and control systems, their subsystems, the subsystem's subsystems, etc. As with any information modeling technique, the practitioner must know when the costs associated with modeling outweigh the benefits.

Another problem with the model, particularly when used in association with an automated tool, is that its use is often considered a "short cut" to requirements engineering. The model does not replace requirements engineering in complex information systems. Rather, it is a disciplined approach to manage the complexity of a system and its requirements. Applied in the context of security engineering, it affords a technique to graphically illustrate and manage the security requirements associated with a system architecture.

6.0 Conclusions

In conclusion, the Zachman Information Systems Architecture framework for systems modeling provides a commonly used technique that can be applied to security policy modeling early in the system requirements definition process. By applying the top three levels of the Zachman hierarchy, it is possible to develop a descriptive security policy in simple English that can be understood by the system consumer organizations. Annotations to the IDEF0 model for classification, source, destination, and data manipulation constraints allow rapid location of possible problem areas before they are designed or implemented in the system architecture. Use of the lower layers of the model provides additional traceability that is highly useful to the Designated Approval Authority as part of the system security certification evidence. As such, it is a valid tool to apply to security policy modeling when developing an information system. Application of the Zachman Model provides a technique to:

- express doctrine oriented security requirements,
- reconcile these requirements with the "as built" security requirements, and
- provide traceability for requirements from specification to implementation.

[5] Sowa, J.F., and Zachman, J.A., "Extending and Formalizing the Framework for Information Systems Architecture," *IBM Systems Journal*, Vol. 31, No. 3, 1992.

Ideally, incorporation of security requirements into the framework should result in a more integrated approach to security requirements analysis, with the eventual inclusion of security requirements engineering into conventional systems engineering as an integrated requirements engineering activity.

7.0 Acknowledgments

The author is grateful to Adele Park, Tony Whalen, Eric Meijer, Mary Englert, and the Harris Information Systems Reengineering Action Team for providing their resources, thoughts and comments to the preparation of this paper.

8.0 References

Bruce, Thomas A., “Simplicity and Complexity in the Zachman Framework,” *Database Advisor*, pp. 3-11, May/June 1992.

Holbein, R., Teufel, S., and Bauknecht, K., “A Formal Security Design Approach for Information Exchange in Organizations,” *Proceedings of IFIP Working Group 11.3 Ninth Annual Working Conference on Database Security*,” Rensselaerville, NY, pp. 291-317, August 13-16, 1995.

IDEF Users Group, *The IDEF Framework*, Version 1.2, IDEF-UG-0001, May 22, 1992.

Mayer, Richard J., Painter, Michael K., deWitte, Paula S., “IDEF Family of Methods for Concurrent Engineering and Business Re-engineering Applications,” Knowledge Based Systems Inc., 1993.

Pickett, R., “Process Modeling through IDEF, a White Paper on Applied Information Technology,” 3 December 1993.

Sadowski, A., et al., *Enterprise Management Analysis*, Enterprise IMS TR4191-01, July 15, 1993.

Schoch, D.J., and Laplante, P.A., “A Real-time Systems Context for the Framework for Information Systems Architecture,” *IBM Systems Journal*, Vol. 34, No. 1, pp. 20-38, 1995.

Sowa, J.F, and Zachman, J.A., “Extending and Formalizing the Framework for Information Systems Architecture,” *IBM Systems Journal*, Vol. 31, No.3, pp. 590-616, 1992.

Zachman, J.A., “A Framework for Information Systems Architecture,” *IBM Systems Journal*, Vol. 26, No. 3, pp. 276-292, 1987.