Proceedings of the 7th IEEE Computer Security Foundations Workshop, Franconia, NH, June 1994, pages 66-79

Conceptual Foundations for a Model of Task-based Authorizations*

Roshan K. Thomas and Ravi S. Sandhu†
[6pt]Center for Secure Information Systems
&
Department of Information and
Software Systems Engineering
George Mason University
Fairfax, VA 22030-4444

ABSTRACT

In this paper we describe conceptual foundations to address integrity issues in computerized information systems from the enterprise perspective. Our motivation for this effort stems from the recognition that existing models are formulated at too low a level of abstraction, to be useful for modeling organizational requirements, policy aspects, and internal controls, pertaining to maintenance of integrity in information systems. In particular, these models are primarily concerned with the integrity of internal data components within computer systems, and thus lack the constructs necessary to model enterprise level integrity principles. The starting point in our investigation is the notion of authorization functions and tasks associated with business activities carried out in the enterprise. These functions identify the authorization requirements while the authorization tasks embody the concepts required to carry out such authorizations. We believe a model of task-based authorizations will bridge the existing gap between low-level models and very high level ones looking at integrity from a purely organizational and sociological perspective devoid of any direct links to computerized systems. The work described here is preliminary and conceptual in nature, but is a necessary prerequisite for the eventual development of a formal model.

*To appear in the proceedings of the *IEEE Computer Security Foundations Workshop VII*, June 14-16, 1994, Franconia, New Hampshire. All correspondence should be addressed to Ravi Sandhu, ISSE Department, George Mason University, Fairfax, VA 22030.

†The work of both authors is partially supported by the National Security Agency through contract MDA904-92-C-5140. We are grateful to Nathaniel Macon, Howard Stainer, and Mike Ware for their support and encouragement in making this work possible.

1 Introduction

Over the last two decades, we have continued to witness the computerization of organizational functions and information-related services in modern organizations. As such, information has become the lifeline of many organizations and can be used directly for competitive advantage. These trends have all contributed to information and computer security being a significant concern among information system managers.

While considerable advances have been made over the last few years in the development of computer security models, a retrospective analysis of these developments would reveal that they have not kept up with the emerging needs and paradigms of computing today. In particular, these models reflect a bias towards a centralized notion of computing, and as such, security objectives are couched in terms of the protection of a centralized pool of resources within a computer. In essence, the overriding concern has been the fine-grained protection of individual objects and subjects in the system. The problem with this approach, of course, is that while it may form a reasonable basis for a computer security model, it lacks the concepts and expressiveness of an *information-oriented model* that captures the organizational and distributed aspects of information usage.

In this paper our focus is mostly on the provision and maintenance of integrity in information systems. We are particularly interested in the integrity issue from an enterprise perspective. We recognize that integrity issues and the design of integrity mechanisms, have lately received a great deal of attention among researchers. In this workshop last year, Sandhu summarized the various definitions of integrity that have been reported in the literature [17]. Interest in integrity area seems to have been sparked off by the well-known paper of Clark and Wilson [6]. Many researchers today are in agreement with the central point of the paper which can be stated briefly as follows: *In commercial data processing environments, the primary concern is the assurance of integrity rather than improper disclosure.* Integrity in this context involves the prevention of fraud and errors particularly in the management and accounting of corporate records and assets.

The justification for further investigating integrity issues can be attributed to many observable needs and trends in computing today. In particular, the increased automation of organizational functions and workflows, and the subsequent need to computerize information systems that often have distributed processing needs. Increased automation always carries it with the risk of weakened controls, especially when human judgement and paper-based checks and balances are taken out of the loop. The emergence of multisystem applications and information-related services that cross departmental and organizational boundaries, call for modeling constructs and integrity mechanisms beyond those existing for centralized systems.

Even a cursory look at modern organizations would reveal them as complex webs of activities (tasks) that often span departmental and organizational boundaries. Tasks are authorized and initiated by users in accordance with their roles, responsibilities, and duties (obligations) in the organization. One can view an organization as a system that is required to maintain a certain state (or standard) of integrity. Organizational procedures and internal controls then have to ensure that the tasks carried out in the organization preserve such a state of integrity. Now when we computerize organizational functions, we are faced with the problem of maintaining the required integrity in our computer-based information systems.

In light of the above, we advance in this paper, the notion of task-based authorizations (TBA) first introduced by the authors in [19]. Task-based authorizations are concerned with the modeling and management of the authorizations of tasks (activities) in information systems. Our obvious objective is the preservation of integrity. This is because unauthorized activities lead to the unauthorized modification of information which in turn affects the integrity of the system. In a paper-based system, authorizations manifest as signatures on documents propagating through the organization. The analog to this in a computerized information system would be digital signatures on electronic documents. As such, we believe that task-based authorizations are central to the successful evolution of the concept of the "paperless office".

The effort described in this paper is by no means meant to be complete or comprehensive. The primary objective is to present our preliminary ideas so as to stir up discussion on richer integrity models. We consciously resist the temptation to prematurely formalize the concepts or build a formal model. We anticipate that this line of work will eventually lead to a formal model. However, we must first develop the conceptual foundations for such a model.

The rest of this paper is organized as follows. The next section covers some background material on security requirements modeling and paper-based internal controls. Section 3 motivates the notion of authorization functions and task-based authorizations by way of an example and presents some of the issues involved in developing a model. Section 4 introduces various modeling constructs for specifying task-based authorizations and illustrates their use by modeling an application. Finally, section 5 concludes the paper.

2 Background

In this section we give the necessary background required to understand the scope, as well as the issues, addressed in the paper. We begin by discussing the different levels of abstraction that are available in approaching security requirements and models. This is followed by a discussion of paper-based controls.

2.1 Security Requirements and Models

One can view the security requirements of a system at different levels of abstraction. In a paper presented at the Computer Security Foundations Workshop in 1991 [12], LaPadula and Williams gave a useful layered taxonomy of stages, where the security requirements at higher stages are successively refined and elaborated at lower stages. Starting with the highest stage, these include:

1. *Trust Objectives:* The basic objectives to be achieved by a system.
2. *External-Interface Requirement:* This specifies the system's x interface to the environment, in terms of the security requirements.
3. *Internal Requirements:* Specifies requirements that must hold within the components internal to a system.
4. *Rules of Operation:* These rules explain how internal requirements are enforced.
5. *Functional Design:* This is a functional description of the behavior of system components.

The security requirements of a system at stages 1 and 2 above, are at a much higher level of abstraction than those at stages 3, 4, and 5. The higher stages specify *what* needs to be done, and these get refined into detailed executable specifications that deal with *how* things are to be done. The higher stages thus involve people-oriented policies and requirements while the lower ones are more computer-oriented.

Given the above stages of elaboration, it is possible to formulate security models for each of these stages, as well as classify existing models as to where they belong. In fact, it is possible to derive a related taxonomy of security models for the above stages (see figure 1). At the highest level we have models to capture organizational policy and requirements that pertain to security. These requirements are then applied to the interface between the organization and the computer system and captured by computer policy models. Computer policy models in turn are implemented by access control models, which in turn map to implementation models, and so on.

As observed in [7, 8] most research and development in security models have been primarily aimed at specifying and implementing internal requirements and related rules of operation. Consequently, there is a mature body of literature on access control and implementation models. The Bell-LaPadula model for multilevel security [4], the HRU model [11], and the typed access matrix model (also called TAM) [18] all fall into this category.

However, models at higher stages are fewer and much of the research is still at its infancy. In the category of models that capture external-interface requirements, the non-interference model proposed by Goguen and Meseger in [10] was the first fully defined model. The original formulation of non-interference was in the context of

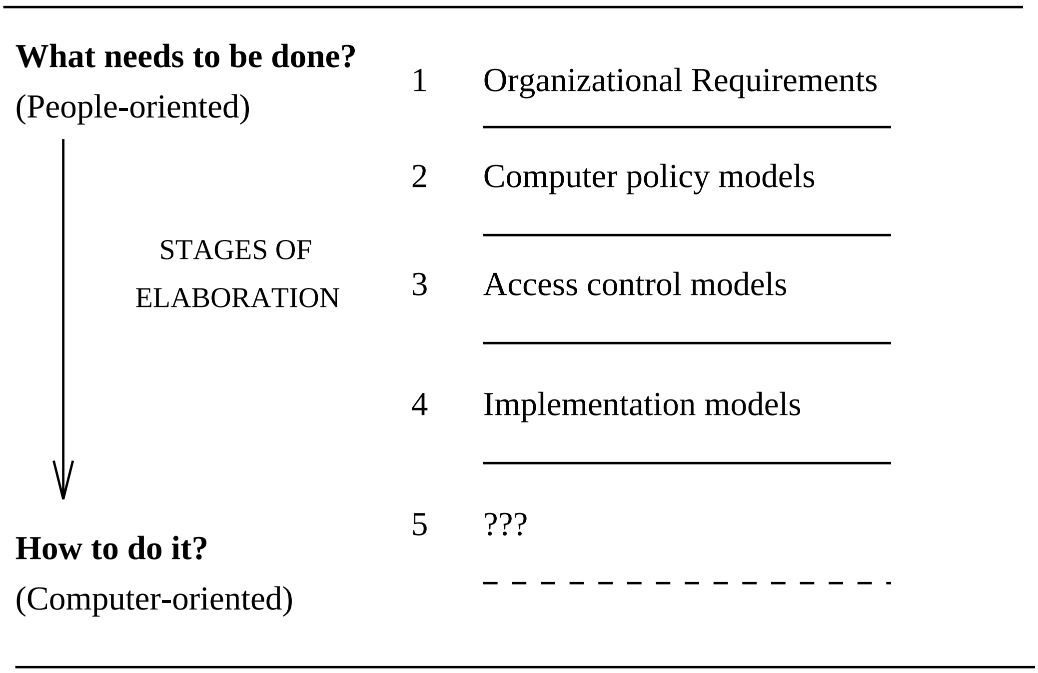


Figure 1: A taxonomy of models

deterministic systems. Subsequently, a number of researchers have developed similar abstract models for information flow in non-deterministic systems [9, 13, 14].

Security models for the first stage of elaboration in the taxonomy of figure 1 are by far the least developed and perhaps the most crucial. Some promising initial efforts have been reported in [7, 8]. The starting point for this approach is the analysis of the various responsibilities and obligations in the organization. This would lead to a better understanding and account of the authorization functions and structures in the organization.

Given such a taxonomy of security models, where would a model of task-based authorizations fit in? We argue that task-based authorizations are an attempt to formulate integrity models to bridge the gap between the internal requirements and higher stages of elaboration. We would like to think of our approach as one that lies above the internal-requirements stage, and perhaps approaching the second-stage of elaboration; i.e., that of external-interface requirements. Hence a model of task-based authorizations falls under the category of computer policy models shown in figure 1.

In concluding this section we briefly discuss the limitations of existing integrity models. The model of Biba [5] is essentially a lattice-based model of information flow. Such a model is clearly inadequate to express complex integrity requirements. The model of Clark and Wilson [6] utilizes the notions of transformation procedures (TP's)

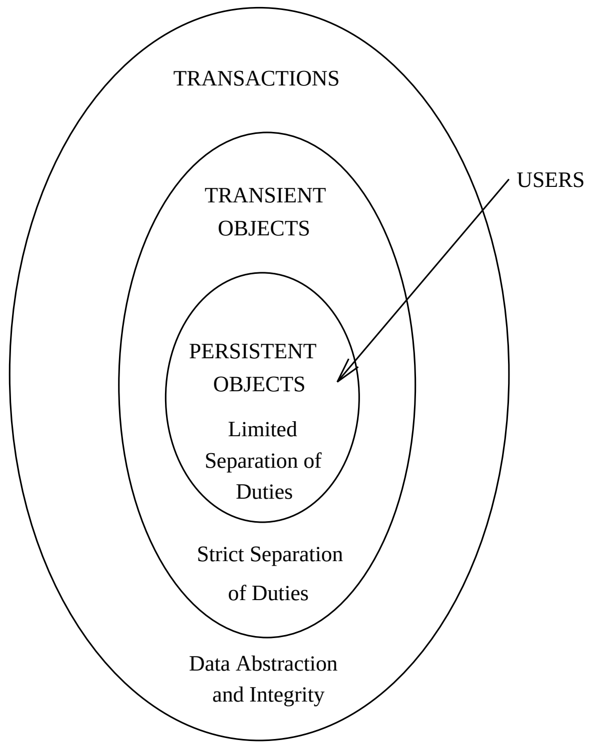


Figure 2: Illustrating an activity model of paper-based controls

or transactions, and constrained data items (CDI's). However these are too low level abstractions. Thus we cannot, for example, capture integrity policies which call for sequences of TP's. The model also suffers from the drawback of mixing several levels of abstraction. Thus we have TP's and CDI's which are computer-based abstractions alongside notions of verification which require organizational and user involvement.

The model defined by Badger in [3] is an attempt to improve on the limitations of the Clark-Wilson model, and recognizes that integrity policies occur at multiple granularities. In particular it is able to express more semantics and structure (such as that of nested transactions) among transformation procedures. Finally the model of Sandhu [15, 16] based on transaction control expressions is able to capture linear sequences of transformation procedures. However, it also has additional abstractions to model paper-based internal controls. All the models above suffer from a bias towards mechanism-oriented abstractions. The work reported in this paper is an attempt to generalize, as well as transcend, all these models by seeking more specification-oriented (as opposed to mechanism-oriented) constructs.

2.2 Paper-based Controls

In any organization, the task of counteracting risks involved in doing business falls under the purview of internal controls. Examples of such risks include fraud, sabotage, embezzlement, to name a few. In addressing the integrity issue in our context, we are particularly interested in controls that safeguard corporate assets and preserve the integrity of accounting data. These include controls on the authorization, creation, and execution of transactions, requirements for separation of duties, procedures for recording and processing transactions, procedures for verifying the accuracy of data collected, to name a few. The accounting profession recognizes many other accounting and auditing principles. Details of these can be found in publications such as [20].

In exploring integrity models, it is helpful to look at paper-based controls in manual systems. In such systems transactions are initially captured on source documents. Examples of such documents include deposit slips and sales order forms. These documents can be used to implement many controls. Consider for example the authorization of transactions. This is achieved in general on a case-by-case basis by requiring each source document to be authorized. The authorization itself manifests in the form of entries such as signatures and authorization codes. Source documents also provide useful information for the verification of data, for constructing audit trails, and for recovery in the event that processed data is lost.

Unlike their manual counterparts, transaction processing in computerized information systems often do not involve source documents in the loop. For example, with electronic data capture, many systems bypass source documents. This can lead to the weakening of many controls. Thus in order to get a grip on the many integrity issues, we believe it is a good idea to apply the principles of internal paper-based controls in computerized systems. The work reported in [15, 16] is an attempt in this direction and presents a model and notation for this purpose. The intuition behind this approach is illustrated in figure 2 and centers around the notions of *transient* and *persistent* objects. Transient objects include documents such as vouchers, purchase orders, sales slips, to name a few. These objects are transient in nature in the sense that they issue a finite set of operations and then leave the system (in a paper world this happens when a form is archived). These operations eventually affect persistent objects such as inventory databases, and bank accounts. The fundamental idea is to enforce controls primarily on the transient objects, and for transactions to be executed on persistent objects only as a side effect of executing transactions on transient objects.

To incorporate internal controls such as separation of duties, the model introduces the notation of *transaction control expressions*. Consider a check processing application where a clerk has to prepare a check and assign an account, followed by three (separate) supervisors who have to approve the check and account, and finally the check is to be issued by a different clerk (in the paper world, this would be accomplished trough a voucher). This can be represented by the following transaction control expressions:

prepare • clerk;
3: approve • supervisor;
issue • clerk;

The colon is a voting constraint specifying 3 votes from 3 different supervisors. Each expression consists of a *transaction* and a *role*. Separation of duties is achieved by requiring the users who execute different transactions in the transaction control expression be all distinct.

In summary transaction control expressions are a good starting point in attempting to mimic paper-based controls. They provide support for linear sequences of authorizations, and for controls based on separation of duties and multiple approvals.

3 Task-based Authorizations

In the last section we motivated the need for more abstract models to capture integrity requirements at the enterprise level and discussed the usefulness of paper-based control principles in computerized information systems. With that background, we now turn our attention to the central theme of this paper, the modeling and management of task-based authorizations.

3.1 Authorization Functions and Tasks

As mentioned earlier, task-based authorizations are concerned with the management of authorizations of activities. The need for authorizations arises from the existence of *authorization functions* alongside business activities carried out in the enterprise. In fact, to be more precise, task-based authorizations are concerned with the execution and management of authorization functions. To illustrate the role of these functions, let us consider the classical example of sales-order processing. The document flow for such an application is shown in figure 3.

Processing is initiated with the receipt of a requisition order at the requisition office of the university. The authorization function here involves verification and authorization of the details (terms) of the requisition such as the quantity ordered, price per unit, vendor, and availability of funds. After this step, the requisition order is sent to the sales department of the associated vendor(s). In this process the requisition activity now crosses organizational boundaries. On receipt of the requisition order, the sales department now has to generate a sales-order. The authorization function now has to do with the terms of the sales. This may include among other things negotiation and approval of discounts and delivery dates. The sales-order document then propagates through several departments in the vendor organization. Each de-

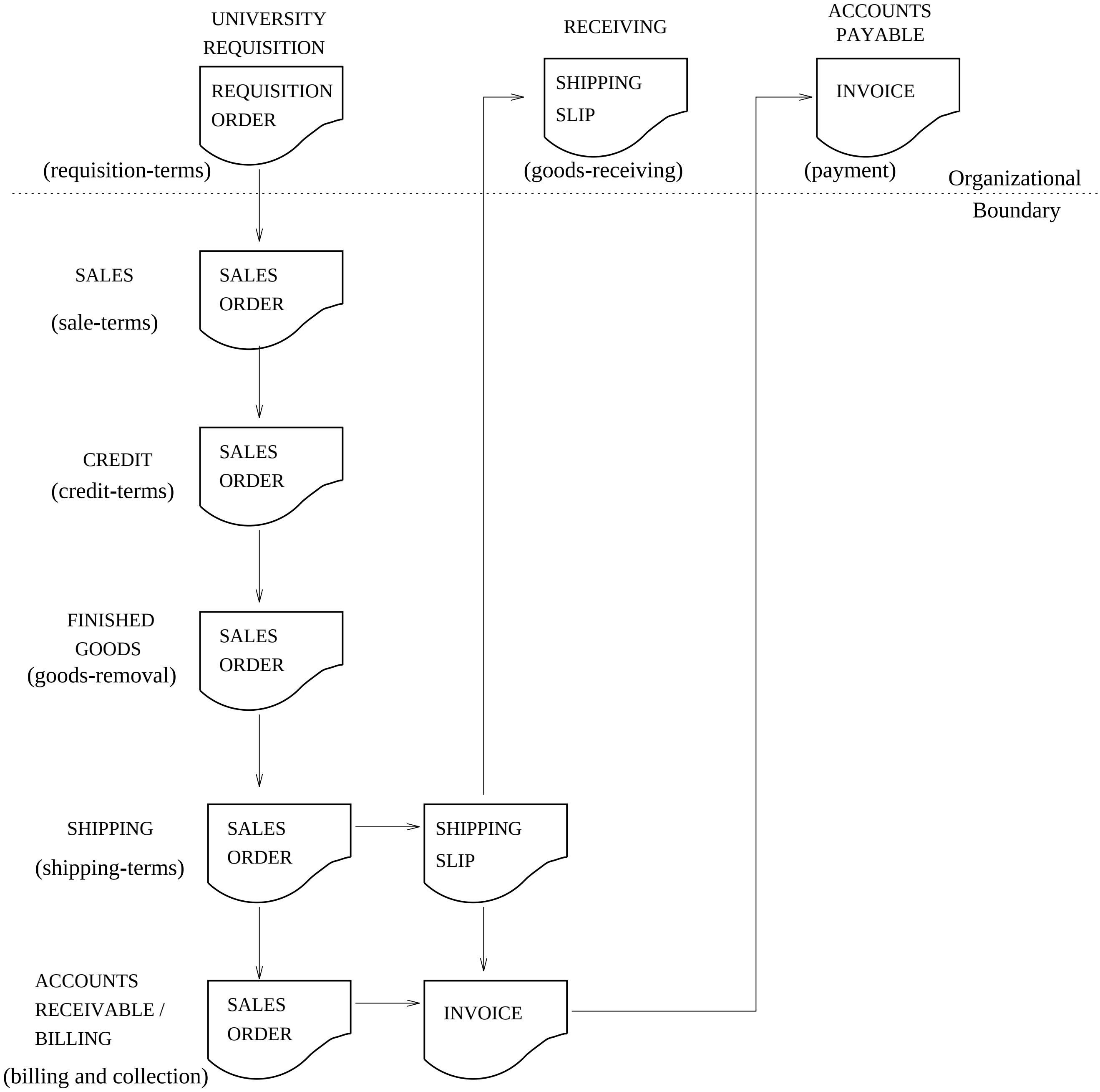


Figure 3: Authorization functions in a sales-processing application

partment involves a different authorization function. Thus the credit department undertakes a credit check and approves (or disapproves) of the credit rating of the customer, followed by the finished goods department's approval of the removal of goods from the warehouse, which in turn is followed by the authorization at the shipping department to ship the goods transferred from the warehouse. Finally the billing department is authorized to collect payment and the receiving department authorized to receive/collect the shipped goods for delivery to the customer who originated the requisition.

An understanding of the interaction between transaction cycles and authorization functions is crucial to building a model of task-based authorizations. In particular, the outcome of authorization functions have direct consequences on the completion of transactions as dictated by policy. Thus a vendor may decide not to continue with a sales order transaction if the authorization function returns a poor credit rating for the customer, since doing so would involve taking risks that may affect future cash flows. In addition, authorization functions may be of varying complexity. In a paper world, the simplest case would be an authorization that requires a single signature. More complex forms may require multiple approvals (signatures), such as when approval from multiple warehouse managers are required to transfer all goods to the shipping department.

It is worthwhile to see how the above ideas map to the modeling of responsibilities and obligations in the framework of [7, 8]. In examining the sales-processing application, we observe that the outcome of the authorization functions creates a network of responsibilities and obligations as mentioned in [7, 8]. Thus the authorization of the sales-order makes the sales department "responsible" for fulfilling the sales-order. This indeed is the main function of such a department. However, to fulfill this responsibility the sales-department or sales-agent may transfer associated obligations to other agents. These obligations create new responsibilities. Thus the finished-goods department takes responsibility for the goods removed from the warehouse, while the shipping department accepts responsibility for the condition and safety of the shipped goods.

In summary, each authorization function is a point in a network where responsibilities are accepted, and one from which associated obligations are discharged and new responsibilities created. Thus authorization functions could be the proper abstraction or boundary object that glues an organizational model based on responsibilities to a more computer-oriented model (at the second stage of elaboration) such as task-based authorizations.

3.2 Issues

In this section we consider the numerous issues that arise in the development of a model of task-based authorizations.

- **Abstraction and Composition**
  One of the first issues that arises is that of abstraction and modeling. What is the proper abstraction to specify and manage authorization functions and tasks. We propose the abstraction of an *authorization-task-unit* to model the authorizations associated with every authorization function. Such an authorization unit may be composed of other smaller units called called *approval-steps* (this is analogous to the composition of functions). These smaller units map to the individual approval steps required to complete the processing of an authorization function.

- **Dependencies**
  Any model of authorizations must have the expressive power to model the dependencies between authorization-units as well as those internal to an authorization unit. These dependencies arise due to the structural and semantic properties of the responsibilities and activities in the enterprise. There exists various kinds of dependencies. Some of these are identified below.

  - *Temporal*
    Here we are concerned with dependencies that constrain the temporal order of the execution of authorizations. Consider an application that requires three approvals (signatures) S_1, S_2, S_3. Organizational policy may require the following dependencies to be enforced:
    (a) S_2 cannot be granted until S_1 has been granted;
    (b) S_3 cannot be granted until both S_2 and S_3 have been granted.

    In our sales processing application, the sales-order is allowed to progress only after the credit manager in the credit department signs off on the order.

    It is clear that we have to address issues related to both the specification and enforcement of dependencies.

  - *Semantic*
    Here we are interested in dependencies that arise from the semantics of the application. For example, seeking authorization to transfer funds between two accounts may semantically imply the need for authorizations to withdraw from the source account as well as deposit in the target account. How can such a semantic unit be expressed and managed?

  - *Atomic*
    We may require the granting of a certain group of authorizations to be atomic. In other words, if one of the authorizations in a group is not granted, we may wish that others in the group to be not granted as well since we want the system to be unaffected by the entire group. The atomicity requirement may directly follow from the semantics of the application,

and its implementation may require interactions with revocation mechanisms. Is there an analog to the atomic transaction in the realm of authorizations? One could think of the abstraction of an *atomic-authorization-task-unit* that guarantees atomicity of authorizations internal to it.

- **Incorporation of controls**
  What are the proper constructs and mechanisms needed to specify and enforce internal controls such as separation of duties, multiple approvals, and rotation of assignments? A general model must support such controls both within, as well as across, authorization-task-units.

- **Delegation and revocation**
  In our sales-processing example, the vendor organization might upgrade the credit rating required of its customers, and as a result, the credit authorization on a sales order may be revoked if a customer fails to meet this new cutoff. In other words, the credit manager is now no longer willing to take responsibility for such a customer. Examples of this call for appropriate delegation and revocation mechanisms.

- **Authorization expirations**
  In the paper world, a signature on a form has validity only for a certain period. In other words, the authorization has an expiration date. If an authorization expires, the related activities may have to be cancelled, and other authorizations whose validity is conditional on the expired authorization, may have to be revoked. We are thus faced with issues related to the modeling and implementation of expirations.

- **Authorization deadlines**
  There exists scenarios in organizations where an authorization may have to be obtained within a deadline. For example, a manager responsible for giving approvals may be available only for certain hours during the week, or may be going on vacation for the next two weeks. In this case, we may want to associate deadlines for the obtaining of authorizations so as to meet customer needs in time. Such deadlines will in turn directly impact the scheduling priorities of authorization-tasks.

- **Failure and Exception handling**
  If a certain authorization is not forthcoming, how do we specify alternate authorizations? Also how do we specify exceptions to general policy? For example, a new customer may not have any established credit and the organizational policy may call for the approval of the customer's sales order so long as it does not exceed a certain amount. Another example is when a manager is unavailable, and we wish to specify that someone else be allowed to authorize on the manager's behalf.

- **Deadlocked authorizations**
  Is it possible for authorization-tasks to become deadlocked? If this happens, does it imply that there is something wrong with the authorization and responsibility structures of the organization?

The above list is not meant to be a complete one, but rather to be indicative of the complexity involved in formulating and implementing a model. It should also be clear that some of the issues listed are related to specification and modeling, while others (such as deadlocks) are related to implementation.

In comparing the above list to transaction control expressions (TCE's) [15, 16] and Badger's model [3], we see that they provide support to express limited dependencies. Thus TCE's can express only linear dependencies while Badger's model can express nested (hierarchical) structures. TCE's also provide separation of duties only within individual transient objects. It is not clear how Badger's model can be linked to enterprise level requirements and policies. Neither of these models provide constructs to express authorization deadlines and expirations.

4 Groundwork for Building a Model

In this section we develop the basic building blocks required to construct a model of task-based authorizations. Our purpose is not to introduce a formal model (or the machinery for this) as doing so would be premature at this point. It is also important to bear in mind that we are not describing mechanisms (the how), rather the concepts (the what) for which mechanisms would have to built later.

4.1 Basic Modeling Constructs

The basic modeling constructs in our model are listed below. An application is built using *authorization-task-units* which in turn are composed of individual *approval-steps*. The various task-units and approval-steps in an application are related to each other through *dependency specifications*.

- **Authorization-task-unit(task-name):**
  Each authorization task contains the following fields:
  - Originating-function:*function-name*
  - Attributes: *Atomic, Expiration, Deadline*
  - Dependency Specifications:{ }
  - Approval-steps:{ }

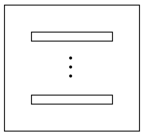
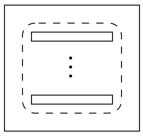
Constructs	Graphical Representation	Explanation
Structural :		
Auth-task-unit		An authorization-task-unit maps to an authorization function and consists of one or more individual approval steps.
Atomic-auth-task-unit		This is an authorization-task-unit with the requirement that the granting of all approval-steps defined within it be atomic.
Approval-step		This is the most primitive construct in the model and represents an individual signature/approval step
Behavioral :		
Temporal-dependency	T1 ⟶ T2	Authorization/approval T2 cannot be granted until T1 has been successfully granted
Failure-dependency	T1 —×⟶ T2	T2 can be granted only ofter the failure of T1
Revoke/delegate-on-failure	{ ***r,d*** } T1 —×⟶ T2	r- T2 and all dependent authorizations are revoked d- T1 upon failure delegates its authorizations to T2
Separation-dependency	T1 <- - - - - -> T2	Separation of duties across T1 and T2
Separation-with-role-substitution	***h*** T1 - - - - - - -> T2	Separation with hierarchical role substitution

Figure 4: Graphical illustration of the various modeling constructs

- **Approval-step**
  Each approval step is a tuple of the form:
  - *(step-name, role, expiration, deadline)*
    where *role* is of type R, and $R \in$ Role-lattice
- **Dependency-specification**
  Each dependency specification is a tuple of one the following forms:
  - *(task-name, dependency-type)*
  - *(step-name, dependency-type)*

Authorization-task-units and approval-steps represent the structural constructs in our model. An approval-step represents the most primitive authorization unit. In a manual paper-based system, an approval-step would map to a single signature on a form. Of course, the required authorization for an activity (task) may require several approval-steps just as in the paper world where multiple signatures may be required to authorize a certain task. From the structural viewpoint, the ability to compose individual approval-steps into bigger units such as authorization-task-units is crucial to modeling many realistic authorization functions. One may also argue for the need for structural units bigger authorization-task-units. This may be required to model groups of authorization functions or top-level tasks.

The behavioral constructs in the model are centered around dependencies. Dependencies specify and constrain the execution and behavioral characteristics of the various authorization-tasks. There exist several different types of dependency constructs. These model among other things the temporal order in which authorizations are to be processed, failure and exception handling semantics, revocation, delegation, and separation of duties requirements.

Figure 4 illustrates graphically the various constructs mentioned above. In the next subsection we illustrate how these constructs can be put together to model authorization aspects of the tasks in an application.

4.2 Modeling the Sales-processing Application

We now revisit the sales-order example in figure 3. We illustrate in 5 how the various constructs can be put together to build an application model of task-based authorizations. We examine each authorization function in figure 3 and discuss below how the various modeling constructs are used.

Sale-terms

This authorization-task-unit is charged with the authorization of the terms of the sale and involves two approval-steps, namely, pricing and delivery-date. There is

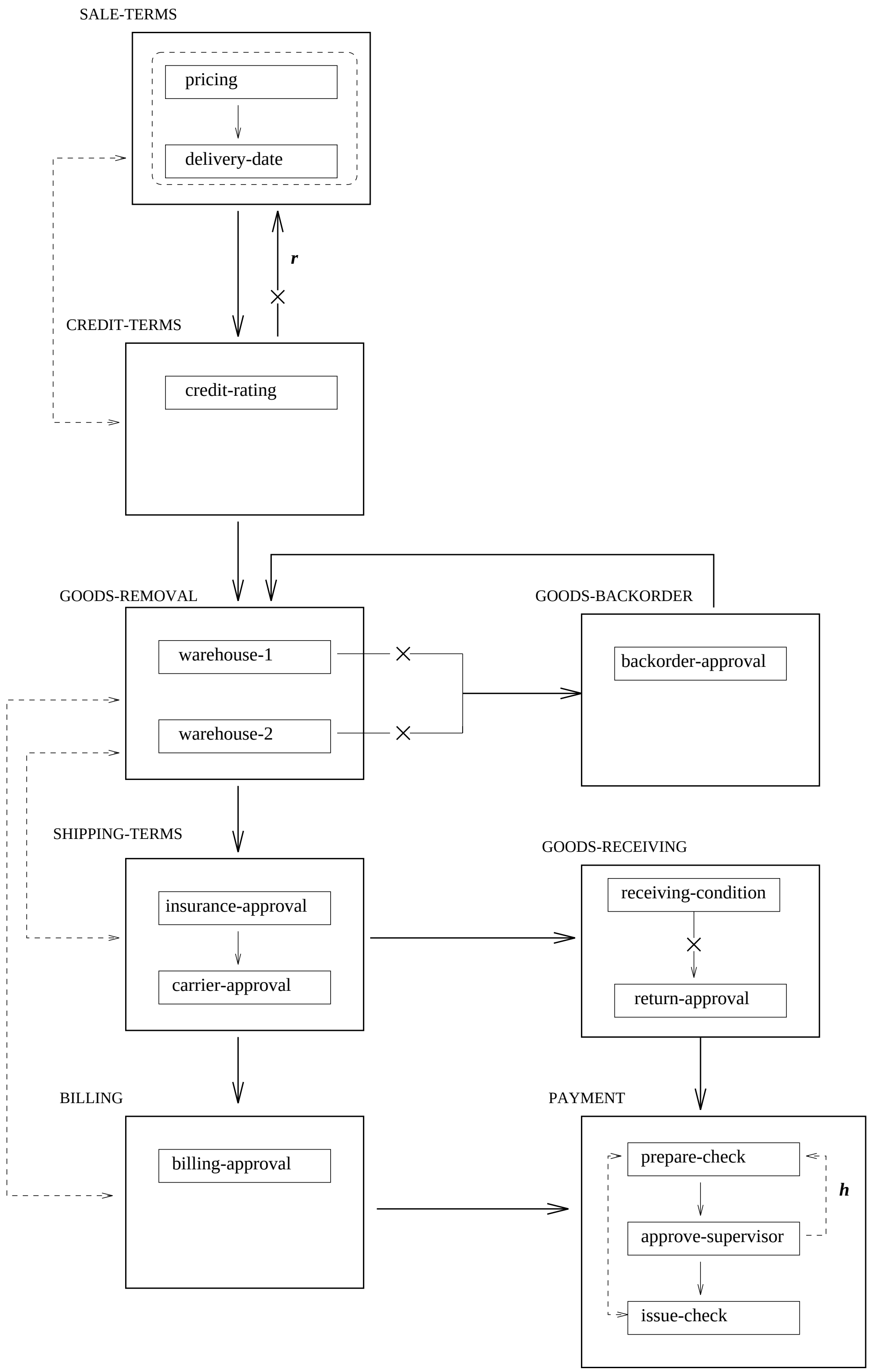


Figure 5: Modeling the sales application with task-based authorizations

a temporal dependency from pricing to delivery-date implying that the latter approval can be granted only after the former (this makes sense as price is the first point of negotiation in many sales). However, both approval-steps are declared in an atomic-authorization-task. Thus failure to receive an approval on either the pricing or delivery-date steps would result in the entire sales-order not getting authorized. Finally we note the temporal dependency from this task-unit to the credit-terms task-unit. In other words, once the sales-order has been accepted, the next activity involves the authorization of the terms to extend credit to the customer.

Credit-terms

This task-unit essentially involves the approval of the credit-rating of the customer. If credit-check succeeds the next authorization-task, namely goods-removal, as indicated by the temporal dependency emanating from credit-terms, is activated. But what if the customer's credit-check fails? In this case a failure-dependency specifies the next course of action. In particular, the authorization granted to the sales-order in the previous task-unit is revoked. This models the fact that the customer's order has the potential to introduce unnecessary risk to the enterprise, since unpaid bills can affect future cash-flows. Finally, the separation of duties requirement is specified across the sale-terms and credit-terms task-units with a separation-dependency. This ensures that a sales-clerk will not ignore high credit risk customer orders in order to close sales deals.

Goods-removal

If the customer's credit check succeeds, processing continues and goods are authorized to be removed from the existing two warehouses (in our example). It is important to note the absence of any dependency between the approval-steps for each warehouse. Why? This is because, from the integrity perspective we do not care about the order of the individual approval steps for the warehouses. However, if any of these approvals fail because of insufficient quantity of goods in the warehouse to satisfy the sales-order, a failure dependency now seeks approval to backorder goods.

Goods-backorder

As mentioned above, this task-unit is activated when the quantity ordered cannot be met from existing inventory levels in the warehouses. Authorization to backorder goods is an example of an exception-authorization sought after the failure to get a previous authorization. After the authorization for backordered items is received, we later check back with the warehouse(s) once again for authorization to remove the required items.

Shipping-terms

This task-unit seeks the necessary approvals to ship goods transferred from the warehouses. We need to obtain approvals on the insurance terms for the shipment, as well as the selection of the carrier, the latter being dependent on the former. We also specify a separation-dependency between shipping and goods-removal. This control ensures that goods removed from the warehouses are actually shipped, and not stolen by an employee.

Billing

This task simply authorizes the billing of the customer account once goods are shipped.

Goods-receiving

This authorization-task is concerned with authorizations necessary as goods are received on behalf of the customer. The receiving department has to sign-off (approve) of the condition of the goods. If the goods are damaged, then receiving-condition approval-step fails, and an approval-step to return the damaged goods is pursued.

Payment

The payment authorization-task-unit exhibits several characteristics. First of all, authorization activities in this task are begun only after the authorizations in billing and goods-receiving are granted. This can be seen in figure 5 as there are two incoming arrows to the payment task-unit. Within the payment task-unit, there exists three approval steps. There is a need for separation of duties requirements within this task-unit (unlike the others in this application which where across task-units) between the first and the last approval-steps. Thus the prepare-check and issue-check would have to be undertaken by two different clerks. There is also a variation of separation of duties with hierarchical role substitution between the first step (prepare-check) and the second step (approve-supervisor) [16]. In this variation, if no clerks are available to approve the prepare-check step, a supervisor may substitute for a clerk. However, we still need to enforce separation of duties, and the same supervisor will now not be allowed to approve the second step.

It is no means clear, that the constructs given in figure 4 are sufficient to model all combinations of integrity requirements. For example there may be variant of the delegation and revocation constructs outlined here. There also might be conditional dependencies. We hope to investigate these in the future.

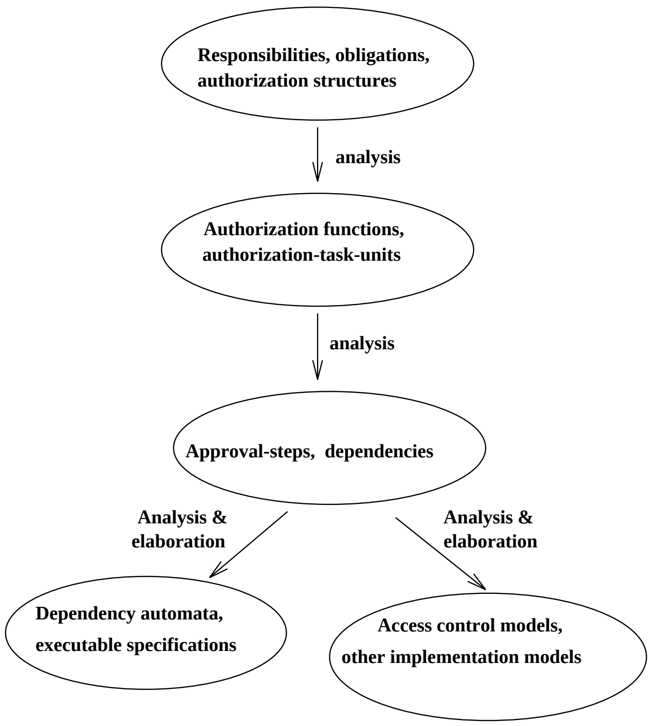


Figure 6: Outline for a design methodology

4.3 Towards a Design Methodology

In this subsection, we briefly outline a design methodology. The basic steps are shown in figure 6. The starting point will be some high-level organizational analysis of the responsibilities, obligations, and authorization structures in the enterprise. Models such as those described in [7, 8] may be useful here. We then derive from this analysis the basic authorization functions and authorization tasks. These are then further analyzed to obtain the internal structure of approval-steps and behavioral structure of dependencies. At this point we have abstract specifications of what needs to be done, in terms of our modeling constructs. We are now faced with elaborating such an abstract specification into an executable one since we want to focus on the enforcement of what has been modeled. This may for example, involve a mapping to access control and other implementation models. The feasibility of this approach was attempted earlier in [1]. Here the authors demonstrated how to implement transaction control expressions using the typed access control matrix model. The enforcement of various dependencies may involve the use of formalisms such as dependency automata [2]. The basic emphasis in this methodology is to start with abstract specifications and to iteratively refine them into executable computer-oriented specifications. We hope to investigate this approach in more detail in the future.

5 Conclusions

In this paper we have laid the groundwork to build a richer model of integrity with the objective of maintaining an enterprise level perspective. By focusing first on the authorization functions associated with business activities, we are able to abstract away unnecessary mechanistic and implementation-oriented details and focus on the authorization structures at the enterprise level. These are then refined and elaborated with necessary constructs to build a computer-oriented integrity model. Integrity in information systems will continue to be a significant issue in computerized information systems. Models such as the one presented here attempt to bridge the gap between very high-level abstract models and extremely low-level computer-based ones. We believe, that the building of such bridges are crucial to advancing the state-of-the-art.

References

[1] P.E. Ammann and R.S. Sandhu. Implementing transaction control expressions by checking for absence of access rights. *Proceedings of the Eight Annual Computer Security Applications Conference*, IEEE Press, December, 1992.

[2] P. Attie et. al. Specifying and enforcing intertask dependencies. MCC Technical Report Carnot-245-92, Microelectronics and Computer Technology Corporation, Austin, TX 78759.

[3] L. Badger. A model for specifying multi-granularity integrity policies. *Proc. of the IEEE Symposium on Security and Privacy*, 1989.

[4] D.E. Bell and L.J. LaPadula. Secure computer systems: Unified exposition and multics interpretation. EDS-TR-75-306, The MITRE Corp., Bedford, MA., March 1976.

[5] K. Biba. Integrity considerations for secure computer systems. *U.S Air Force Electronic Systems Division*, 1977.

[6] D.D. Clark and D.R. Wilson. A comparison of commercial and military security policies. *Proc. of the IEEE Symposium on Security and Privacy*, 1987.

[7] J. Dobson. New Security Paradigms: What other concepts do we need as well. *Proc. of the First New Security Paradigms Workshop,* Little Compton, Rhode Island, IEEE Press, 1993.

[8] R.Strens and J. Dobson. How responsibility modelling leads to security requirements, *Proc. of the Second New Security Paradigms Workshop,* Little Compton, Rhode Island, IEEE Press, 1993.

[9] J. Gray. Probabilistic interference. *In Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy*, IEEE Press, 1990.

[10] J. A. Goguen and J.Meseguer. Security policy and security models. *Proc. IEEE Symp. on Research in Security and Privacy,* Oakland, Calif., May 1982, pages 11-20.

[11] M.H. Harrison, W.L. Ruzzo, and J.D. Ullman. Protection in operating systems. *Communications of the ACM,* 19(8), pages 461–471, 1976.

[12] L.J. LaPadula and J.G. Williams. Toward a Universal Integrity Model. *Proc. of the IEEE Computer Security Foundations Workshop,* Franconia, New Hampshire, IEEE Press, 1991.

[13] D. McCullough. Specifications for multi-level security and a hook-up property. *In Proceedings of the 1987 IEEE Symposium on Research in Security and Privacy*, IEEE Press, 1987.

[14] J. McLean. Security models and information flow. *In Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy*, IEEE Press, 1990.

[15] R.S. Sandhu. Transaction control expressions for separation of duties. *Proc. of the Fourth Computer Security Applications Conference*, pp. 282–286, 1988.

[16] R.S. Sandhu. Separation of duties in computerized information systems. *Database Security IV, Status and Prospects*, S. Jajodia and C.E Landwehr (Editors), Elsevier Science Publishers B.V. (North-Holland)

[17] R.S. Sandhu. On the four definitions of data integrity. *Proc. of the seventh annual IFIP Working Conference on Database Security*, September 12-15, Huntsville, Alabama.

[18] R.S. Sandhu. "The Typed Access Matrix Model." *Proc. IEEE Symposium on Research in Security and Privacy*, Oakland, California, May 1992, pages 122-136.

[19] R.K. Thomas and R.S Sandhu. Towards a task-based paradigm for flexible and adaptable access control in distributed applications, *Proc. of the Second New Security Paradigms Workshop,* Little Compton, Rhode Island, IEEE Press, 1993.

[20] The Auditor's Study and Evaluation of Internal Control in EDP Systems, *American Institute of Certified Public Accountants*, 1977.