Support for Security Modeling in Information Systems Design

Gerhard Steinke
University of Passau, Innstr. 33, 8390 Passau, Germany

Matthias Jarke
Informatik V, RWTH Aachen, Ahornstr. 55, 5100 Aachen, Germany

Abstract

We present a set of modeling constructs and reasoning tools that extend the use of computer-supported conceptual modeling for information systems to the study of security aspects. The modeling framework is the Group Security Model (GSM) which describes access rights through a teamwork-oriented organizational model. Reasoning about GSM application models is enabled by representing them in a deductive and object-oriented database language, Telos. A prototype implementation in the software information system ConceptBase is reported.

1. INTRODUCTION

Conceptual modeling has been widely applied in information systems requirements engineering and design. While this was mostly a pencil-and-paper exercise for quite a while, a number of tools have recently become available through which conceptual models can be formally represented as knowledge bases and analyzed with knowledge base tools [Borgida et al. 87].

Initial conceptual models just specified the internal structure of systems. More recently, the emphasis has shifted to describing the relationship of the information system to its

environment. According to a model proposed in [Jarke 90], this environment can be categorized into the usage, the subject, and the development world of the system. The usage world describes the organization in which the system is intended to function, the subject world describes the part of the world the system maintains information about, and the development world describes the version history of system components and development teams. Each of these subworlds, as well as their interactions, create certain nonfunctional requirements which have to be taken into account when making or evaluating design decisions about the system. Domain theories can be developed which provide conceptual frameworks and reasoning tools for representing and using such requirements.
This paper presents such a domain model for the goal of information systems security in an organizational setting. Security can be understood as a type of nonfunctional requirement which constrains the relationship "access rights" established by the information system between the usage world and the subject world. Our conceptual model, the Group Security Model (GSM), assumes that an information system is embedded in a teamwork-oriented usage environment where teams have to accomplish certain tasks and *therefore* need certain information.

The GSM is a framework for discretionary security policies with some mandatory aspects. It is mandatory in the sense that certain organizational and access right structures are defined in a centralized fashion at information systems design time, and that users may transfer rights only within this framework. However, as in discretionary models, there is no central definition of classification levels for users but a concept of subsystem ownership and distributed control. As a further note to security specialists, we should say that we take a fairly naive view of access rights here: we assume a safe underlying encryption and communication system and ignore the dangers of hidden channels (but see [Steinke 91] for a discussion of one such channel, the handling of integrity constraints).

There are several possible uses of such a model. First, if designers choose the GSM framework for information systems engineering and design, they can use the associated analysis toolkit to study security aspects of the proposed system interactively. Through query mechanisms, they can simulate the system behavior and find out who has access to what objects. Through limited theorem-proving capabilities from the domain of semantic query evaluation, they can study which task (and thus access right) combinations should be avoided from a security standpoint. Through tracing mechanisms, they can obtain explanations how users have obtained or could obtain implicit access to objects.

Similarly, the impact of proposed organizational and design changes in existing systems can be analyzed, and security-oriented re-engineering of existing information systems can be supported when a reverse engineering study of them is made within the GSM framework.

A toolkit for reasoning about GSM-based security is contributed by representing the GSM in a knowledge representation language, Telos [Mylopoulos et al. 90], that combines features of deductive and object-oriented database languages. The security modeling primitives (such as classes of access rights) are defined in this model as metalevel deduction rules and integrity constraints which are then compiled using optimization techniques from deductive databases offered by the software information system, ConceptBase [Jeusfeld and Jarke 91, Jeusfeld and Staudt 91].

The GSM was originally proposed for the context of security in arbitrary multi-user knowledge bases. However, due to the computational problems associated with theorem-proving in complex rule systems we expect our tools to be most practically relevant in the formally rather simple object-oriented knowledge bases used to describe information systems requirements and designs. We therefore focus on this aspect here.

Section 2 gives an overview of previous work on reasoning about security. Section 3 describes the GSM framework and section 4 the associated analysis toolkit. Section 5 provides comments on the implementation as well as suggestions for further research.

2 BACKGROUND AND MOTIVATION

Reasoning is the ability to draw conclusions and create information by applying rules and functions on prior knowledge, in addition to the capability to provide an explanation and justification of specific actions and conclusions.

Reasoning about knowledge, based in the philosophical tradition, has become an active area of research in a wide variety of fields ranging from artificial intelligence and cryptography to linguistics and psychology. Reasoning about knowledge is also a basic feature and capability of knowledge base systems. Knowledge base systems create information based on the triggering and evaluation of rules by the control structure. A "characteristic of knowledge-based systems is that they are (ideally) able to reason about their own reasoning process" [Berson and Lunt 87, p.237]. The capability of this reasoning process should be made available to users in order to help them understand and improve the design, implementation and operation of the security component of a knowledge base system.

U-Kuang is a rule-based tool which performs reasoning about the security of UNIX systems [Baldwin 87]. U-Kuang deduces the set of operations directly or indirectly accessible to each user, particularly those operations which can extend a user´s privileges. In order to determine these operations, U-Kuang contains information about UNIX protection modes, files, privileged programs and files they access, etc. U-Kuang

determines the potential operations accessible to users and compares the operations with the specified security policy which contains all users and their allowed user and group privilege. The operations which violate the security policy are reported.

The application of reasoning about security is often considered to be only an optional or nice, rather than essential security component. Although reasoning about security does not provide security in itself, it enables the security of a system to be examined and evaluated. However, for information engineering in organizations, the possibilities for security strongly depend on the organizational design that implies the "need-to-know".

[Glasgow et al. 91] pursue a somewhat similar point of view when they define reasoning about security in terms of the interacting notions of knowledge, obligation and permission. They create a structured framework for reasoning about security based on a modal logic formalism, as well as a language for expressing abstract security policy definitions. However, no practical reasoning tools are readily available for such special-purpose logics.

One major issue in reasoning about security has been complexity. In the discretionary policies we are interested in here, the early results were either negative (undecidability of the access matrix model [Harrison et al. 1976]) or led to overly simplistic models. However, concepts from the database area, in particular the ideas of database schema and of transaction specification, have recently led to more realistic solutions.

The one closest to our approach is the Schematic Protection Model (SPM) by Sandhu [1988]. Security policies can be defined by a meta database schema which defines permitted relationships between models of active components (*subjects*) with each other and with passive *objects*. The complexity of analysing whether -- under a given initial database state -- a given subject could potentially access a certain object then depends mostly on the way how transfer rights (copy flags) can be transfered among subjects. Specifically, Sandhu identifies two sufficient conditions for polynomial complexity of potential access analysis:

(1) *Acyclicity*: if subjects of type *A* create subjects of type *B_A* directly or indirectly, then subjects of type *B* must not create, directly or indirectly, subjects of type *A*. Cycles within the same type are allowed.
(2) *Attenuatio*n: when a subject *a* (of type A) creates another subject *b* (also of type *A)*, then *b* must have equal or less rights than *a.* This implies that *a* can only transfer its own rights at creation time and that any new right *b* receives later, is also given to *a*.

In related work, Sandhu demonstrates that most practically relevant protocols in the literature can be reformulated to satisfy these conditions. In particular, the coverage of hierarchically organized groups such as formal organizations is possible as well as the consideration of different user roles (such as leader and different levels of members).

In summary, Sandhu´s models provide a thorough study of how the relationships between subjects and objects should be constrained such that security analysis becomes practically feasible. The basic idea is shown in figure 1.(a).

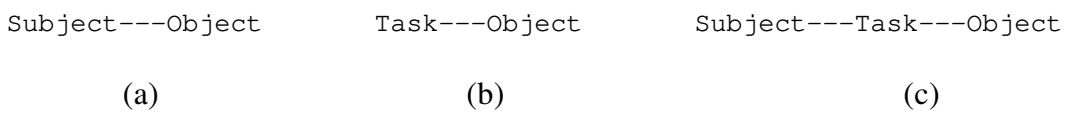

Fig. 1: Security modeling, information systems engineering, and a synthesis.

Unfortunately, this is not exactly how standard methods for organizational information systems analysis and design proceed. Most of them (including, e.g., IBM´s well-known Business Systems Planning) identify the required relationships of business processes or *tasks* to data objects and would ignore the people or groups involved. See figure 1.(b).

Yet, it is individuals (or small groups thereof) who pose the major threats to security; and often these threats stem from the combination of different tasks (and thus justified access rights) addressed by the same organizational unit. The approach described below can be seen as attempting a synthesis between the two viewpoints as shown in fig. 1.(c). From the viewpoint of security analysis, it provides an answer to the question: *where do the subject groups mentioned in security models come from?*

From the viewpoint of information systems analysis, it allows the inclusion of security aspects without much extra data collection work for the analyst. More importantly, it also allows *organizational design for security* in the sense that proposed partitionings of task assignments to organizational units can be analyzed for their security implications. Finally, it should be mentioned that the same conceptual structure also turns out to be useful as a basis for computer-supported cooperative work [Rose et al. 92].

Obviously, to make this idea fully operational, such an organizational model must be mappable to a formal security model such as SPM. In particular, the acyclicity and attenuation properties of a system that contains multiple kinds of interwoven "active components" (subject organizations and task structures) need to be studied. In the following exposition, we do not yet solve this problem completely but offer the following two contributions: a domain analysis of what features and properties the proposed model needs to be useful both from a general information engineering and from a security perspective, and an implemented toolkit that employs deductive query processing facilities for static and partial dynamic security analysis.

3 THE GROUP SECURITY MODEL

The goal of achieving a secure system is approached by restricting a user to a subset of the information in the system. The smaller the subset of the information which a user may access, the less information that can be disclosed or modified by the user, resulting in a more secure system. The smallest subset of information which must be made available to a user consists of that information which the user needs to know. In addition, one can also restrict the operations which a user may carry out w.r.t. the information which he may access. These operations or access types would be defined according to what the user needs to do with the accessible information.

Both the need-to-know and the need-to-do are not just determined by a user´s membership in a formal organizational unit but much more precisely by the tasks he or she is involved in. The Group Security Model (GSM) [Steinke 92] restricts potential user access rights to task-related information while leaving actual access right granting at the discretion of the data or procedure owner.

The GSM is formally defined as a structured semantic network in the deductive and object-oriented database language Telos [Mylopoulos et al. 90]. Telos supports the abstraction principles of classification/ instantiation, generalization/specialization, and aggregation/ attribution. Specifically, the instantiation hierarchy leads to the following layered representation of security-related knowledge (the same is, incidentally, true for other aspects of IS modeling):

- The meta meta class level defines the generic GSM framework.
- The meta level class specifies particular kinds of task-based security policies within the framework.
- The simple class level describes the security model of a particular information system within a meta level policy.
- The instance or token level is a record of actual user roles and accesses within an information system which could be used, e.g., as an audit trail.

Each level can be seen as a type system that provides a sublanguage which constrains what can be defined at the level below. The security analysis toolkit currently focuses on the lowest two levels, making certain assumptions about the policies defined at the meta level.

The two highest levels of the GSM is represented in fig. 2 as a semantic network, using graphical standards of Telos. The upper box describes the meta meta level of the model, the lower one a set of specific meta level constructs we have found useful for our prototype. In the figure, thin links with no labels stand for instantiation relationships, thick grey links for specialization relationships and the labeled links are attributes. Rectangles represent class objects and ovals represent pre-defined simple class objects which are needed in every GSM-based information systems model.

At the metalevel the GSM consists of five basic components: user, role, task, access type and object. The entity `user` represents the individuals who have permission to access the information system. A `task` represents the activities and assignments which are to be carried out in an organization. The entity `object` represents information in the system. A `role` is a link between `user` and `task` which represents the relationship that a user has to a task, e.g., leader and member. An `access type` is a link between `task` and `object` which represents the operations that a user associated with the task may carry out w.r.t. an object, e.g., read, write, and owner.

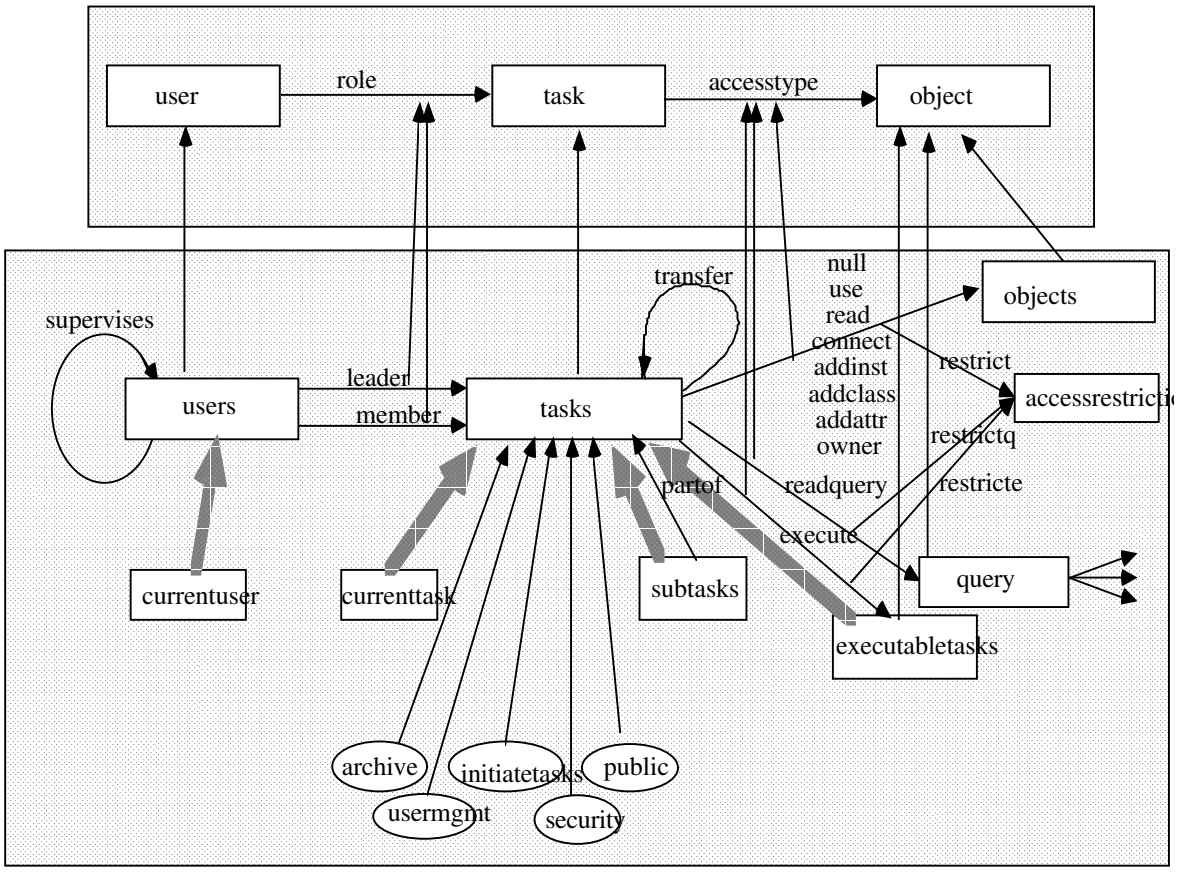

Fig. 2. Group Security Model.

A user is assigned to a task by means of a role link. Information is assigned to a task by means of an access type link. Thereby a user is restricted to that information he needs to access and to those operations he needs to do in order to complete his assigned task.

Besides the above-mentioned abstraction principles, Telos offers deductive rules and integrity constraints as in deductive databases as a basis for formally defining the semantics of concepts. Thus, the GSM and different kinds of policies within the GSM framework can be formally specified by a combination of pre-defined structure (as shown above) with rules and constraints. The following provides an informal overview of the

characteristics and implications of the components of the GSM; the full formal definition can be found in [Steinke 92].

User. The object user in the GSM represents an individual or process which has permission to access the information system. Users may be assigned to more than one task, just as in the real world a user may be required to complete several assignments at the same time. The GSM specifies, though, that a user may only access information associated with a single task at a time, since all information required for a task is associated with that task. Accessing information from several tasks at the same time goes beyond the need to know principle.

The GSM enables supervisor-subordinate relationships between users to be represented by means of the supervises link. The leader of a task may only assign the users which he supervises to his task.

Role. A role defines the relationship which a user has w.r.t. a task. Two types of roles are defined in the GSM, member and leader.

The member role relationship between a user and a task permits a user to access all objects related to the task with the access type specified for that task, except for one restriction. The member role does not permit communication with other tasks, and therefore does not include the right to grant another task access permission to information owned by this task.

The leader role provides a user with responsibilities and priviledges in addition to those granted to a user who is a member of the task. While every task must have a leader, it is possible for a task to have more than one leader, in which case each leader has the same rights and priviledges. Only the leader of a task may communicate with other tasks. A leader is able to grant and revoke access permission to objects owned by this task. A leader has the capability to define subtasks and assign users to his task or to his subtasks.

Task. Task is the central component of the GSM. A task represents an activity or assignment which, in the process of being carried out, requires access to information in the system. There are two types or classes of tasks: first level tasks and subtasks. First level tasks are explicit instances of the class tasks representing tasks which are not subtasks of other tasks, i.e., have no parent task. Subtasks are instances of the subtasks class, created by the leader of a task, and have a partof link to the creating task, i.e., their parent task.

A transfer link, from one task to another task, indicates that access to information which is owned by the former task may be granted to the latter task.

A number of task classes are required by the GSM for administrative purposes. The public task is a task which has associated with it information which all tasks may access.

The initiate tasks task allows a user associated with this task to create and terminate first level tasks. A user with a role in the user mgmt task may add and delete users and supervises links in the GSM. The archive task is the owner of all objects which have no owner, e.g., where the task which was the owner has been terminated. The current task is a subclass of tasks which contains only one instance, namely the task which is currently accessing the information system. A user associated with the security task may use the security analysis toolkit described in the next chapter.

Access Types. The access type component of the GSM, represented by a link from a task to an object, describes the operations that a user associated with the task may perform on the object. There are several ways an access type is created. An owner access type link is granted to the task which creates the object. Other access type links are generated by the owner of an object when granting access permission to his object to another task. In addition, access type links are created by leaders of tasks copying them from a parent task to a subtask. Access permissions may also be revoked by the owner of the object.

Access restriction respresents a constraint which may be associated with an access type. This constraint would describe a condition which must be satisfied before the access type is used to permit access to an object. The access restriction is created by the owner of the affected object.

The number of access types is dependent on the security policy and the knowledge representation language. For Telos-based information system descriptions, the following access types have been defined [Steinke 92]:

> **Null** access type for an object means that no access to this object is permitted. While the absence of an access type link also indicates that access to an object is not permitted, null access type reinforces the denial of access, including the denial of indirect or implied access permission, and overrides all other access types.
>
> **Use** access permission for an object allows the object to be retrieved from the knowledge base for use within the system, but prevents the object from being displayed on an output device. Use access of a rule or constraint permits the rule or constraint to be used for deductions and integrity checking, while the content of the rule or constraint remain hidden from the user.
>
> **Read** access for an object means that the object may be retrieved from the knowledge base. The GSM defines read access for a link object to imply read access to both the source and the destination objects. In Telos this applies to attribute links, instanceof links and isa links.
>
> **Connect** access for an object means that this object may be specified as the destination of a link. If read access was sufficient for an object to become the

destination of a link, then —due to referential integrity of the database — the owner of the destination object may not be able to delete his object, even though the only permission he granted was read access.

Addinst access for an object provides permission to create an instance of that object. The task which creates the instance becomes the owner of the new object.

Addattr access for an object means that one has permission to add attributes to the object. If the attribute is an instance of an attribute category then one requires addinst access to the attribute class object. If the destination of the attribute is a specific object in the knowledge base (i.e., not integer or string), then connect access to that object is required as well.

Addclass access type permits the creation of a subclass, and is therefore only appropriate for a class object. If a subclass is created, additional rights are granted to the owner of the subclass w.r.t the parent classes and their attributes, in order to allow inheritance from the parent classes.

Execute access for an object provides permission to initiate a method or process.

Owner access type indicates the task which is the owner of the object. As owner, a task has the capability to perform any operation on that object. Owner access contains the privilege of granting or revoking access permission for the object, to or from other tasks. The owner may delete his objects.

To enable inheritance, owner access of a class object includes additional rights. The owner of a class object has access permission to objects "below" the class object, e.g., all instances of the class object, all attributes of the instances, as well as all subclasses of the class object (but not the attributes associated with the subclasses). In addition, the owner of a class object may access objects "above" the class object. This allows the owner of a class to use the normally inherited attribute categories as well as recognize that instances of his class are also instances of the parent classes.

Read Query access type from a task to a query object results in access permission to a group of objects, namely all objects which satisfy the specifications of the query, at the time that the access is checked. The read query access permission can only be created by the leader of the task which has owner access to all components of the query definition, since knowledge of the objects specified in the query may be provided to the recipient of the query access. Read access permission is required for the query object itself to be accessed, i.e., to see the makeup of the query.

Objects. The object component of the GSM represents an entity or object in the information system, be it a link, an attribute, a rule, a constraint or any other type of object. Only if an access type link from a task to an object exists, may that object be accessed with the operation specified by the access type link.

Deduced knowledge is derived from the triggering of rules. A rule can only be triggered, i.e., used to generate deduced information, if a user has access to the rule. A rule is triggered on that subset of the knowledge base which a user has permission to access. The deduced information can only be read. Access to deduced objects is possible by others, only as they are able to access and trigger a rule which creates the deduced object.

The GSM requires that a rule only be created by a task with owner access to the objects which are components of the predicate that specifies the rule. The reason for this requirement is that when the owner of a rule grants access to the rule, others may possibly discover the components of the rule. If the owner of the rule is also the owner of the objects used in the specification of the rule, then, since he is the owner, he has the right to let others discover these components.

Constraints are similar to rules since in their specification they may refer to various objects in the knowledge base. These components may become accessible to a task which receives access to the constraint. Therefore, only an owner of all components of a constraint can create a constraint in the first place. A constraint only applies to a user if the user has access to the constraint.

4 A RULE-BASED SECURITY ANALYSIS TOOLKIT

The relationship between the GSM and an implemented information system can be viewed as pictured in fig. 3. An access request to the information system goes to the GSM. If access permission for the current user and task is permitted, then data from the information system can be retrieved or modified according to the access type specified. However, since the GSM-based description of the relationships between users, tasks, and objects is a formal Telos knowledge base, the deductive querying and integrity checking facilities of a Telos implementation can be used for analyzing the information systems model in general at the class level, and the actual situation at the instance level of the model. In this section, we describe the rationale and implementation of such a toolkit we have developed for the Telos-based software information system, ConceptBase. As mentioned earlier, a more general environment which covers the meta class level of basic policy specifications as well is still under development. We are working on this based

along two threads: one is based on a mapping of GSM variants to SPM [Sandhu 88], the other on the use of model-generation algorithms such as SATCHMO [Bry et al. 88].

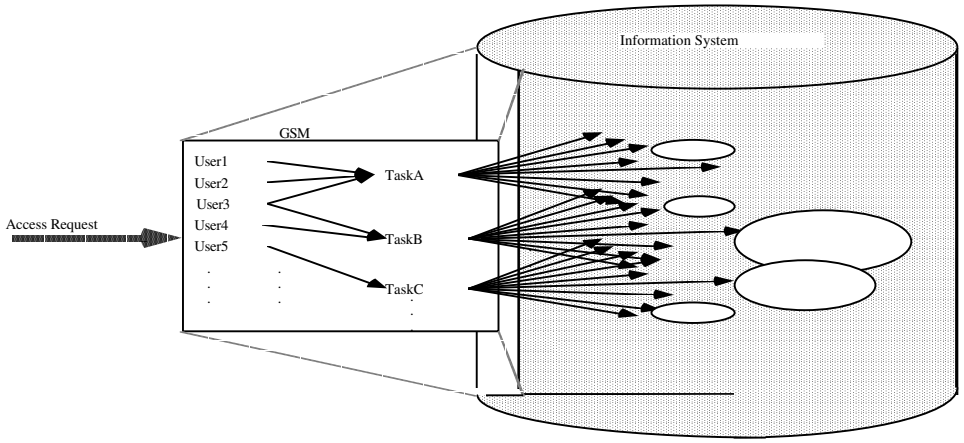

Fig. 3. Access request arrives via the GSM to the information system.

The potential applications of such a toolkit are manifold. Reasoning about security can be used to assess the impact of specific sets of knowledge classifications to determine appropriate security criteria. "Because its knowledge is encoded for logical processing, a knowledge base system is potentially able to reason about the closure of subsets of its knowledge base with respect to inference" [Garvey and Lunt 91, p.34]. Thereby one can find risky situations where insufficient security criteria have been specified for information, or costly situations where information has been over-classified.

Reasoning about security enables a user to look back and explain or justify prior actions in a system. Such a capability is particularly needed by an auditor who must be able to determine who was permitted to access information and why that permission existed.

Reasoning about security enables a preview of the potential impact of a user or an operation. What information can the user already access? To whom may the user pass on information? What new rights could a user possibly get by exercising the current rights? Is there a way a user can get access to a particular object?

Security and access control is required when integrating data from several organizations. This situation arises at the United States Environmental Protection Agency (EPA). A law, "The Toxic Substances Control Act", requires chemical manufacturers to submit details of their products to the EPA. The EPA is required to prevent the disclosure of this data since the information could be used to discover details about corporate strategies, pricing policies, research efforts and development plans. At the same time, "The Freedom of Information Act" requires non-sensitive information to be made public.

In the broader context of the four information systems engineering worlds mentioned in the introduction to this paper, the toolkit could also use reasoning about security to evaluate the privacy implications and requirements of the information in the knowledge base. What permission must be granted from the subject of the information? How do the security criteria relate to the privacy requirements?

In our study, we have used the example of a hospital information system which requires a significant security component to ensure privacy and correctness of its information. Fig. 4 presents a subset of the information contained in a hospital system whereas fig. 5 shows a number of tasks in the hospital environment.

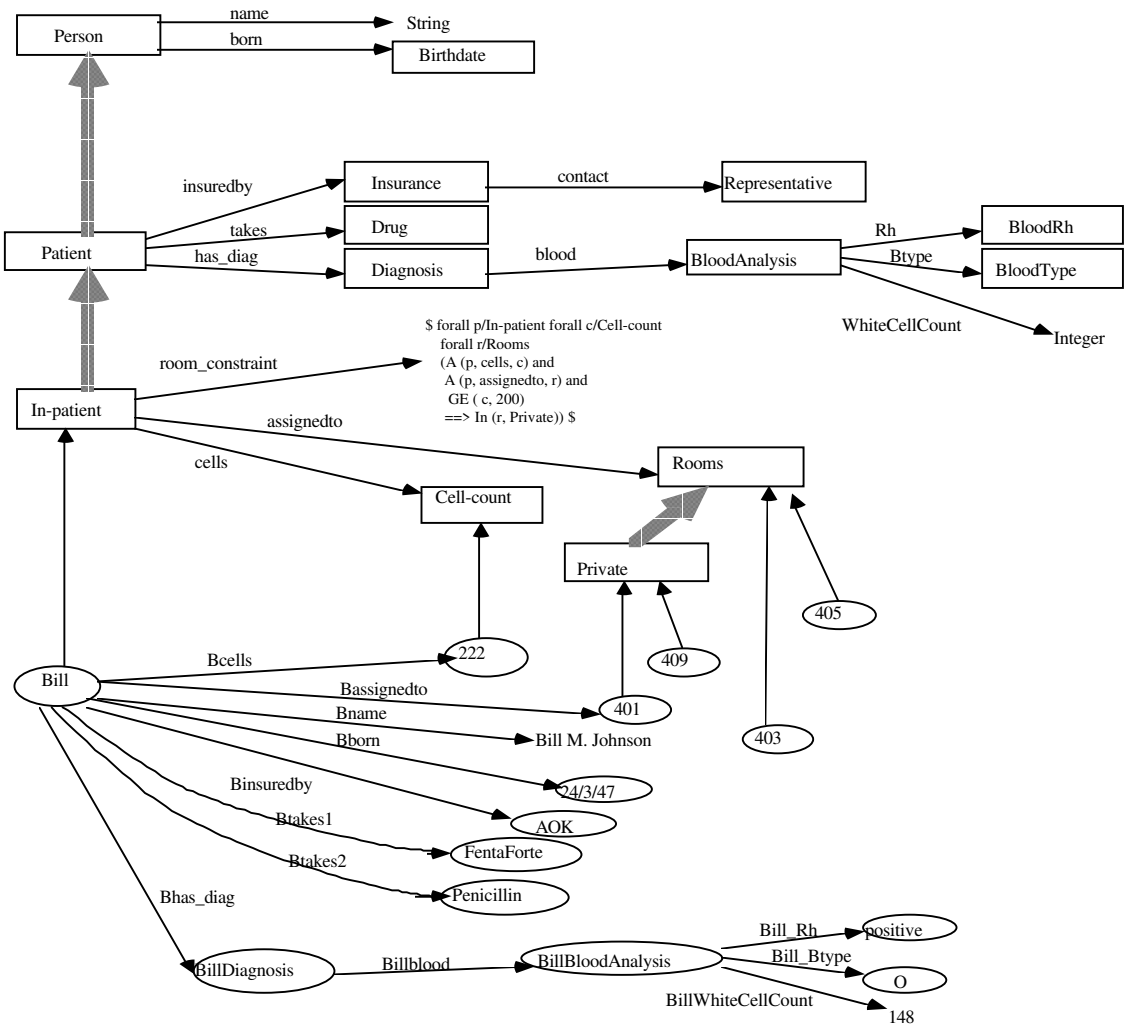

Fig. 4. Telos Hospital Example

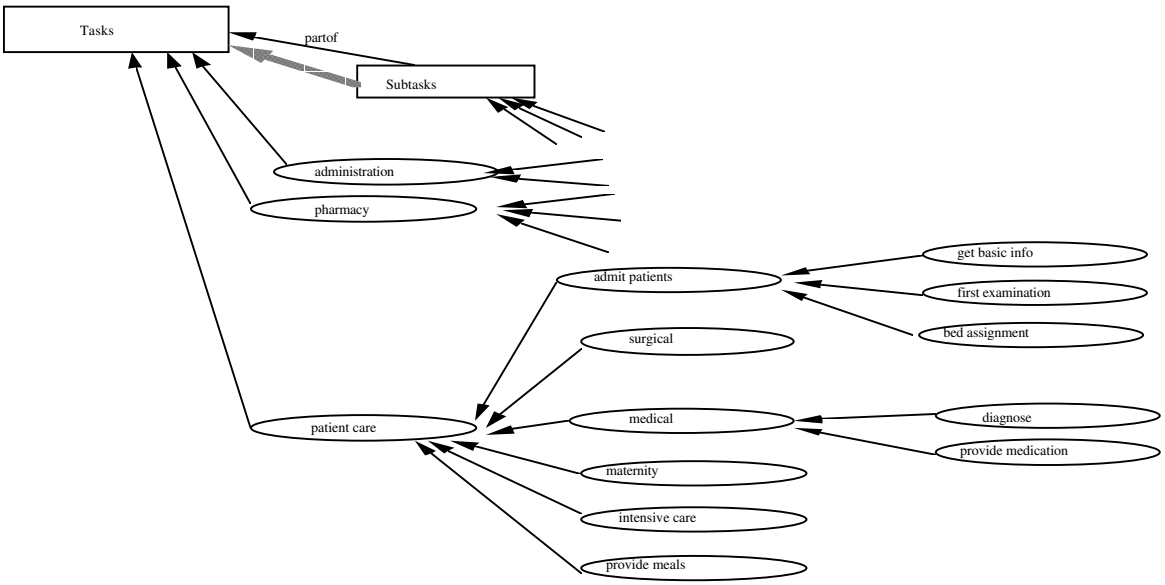

Fig. 5. Tasks in a Hospital environment.

4.1 Toolkit Overview

The interface of the security analysis tool is shown in fig. 6. Users are able to use the security analysis tool according to the nature and requirements of their task. A user of the security task may specify any class level object. That means that the security task may know the structure of the information and who may access information at the class level. Specific instances of information are only available to the owners of the related classes and the owners of the information itself.

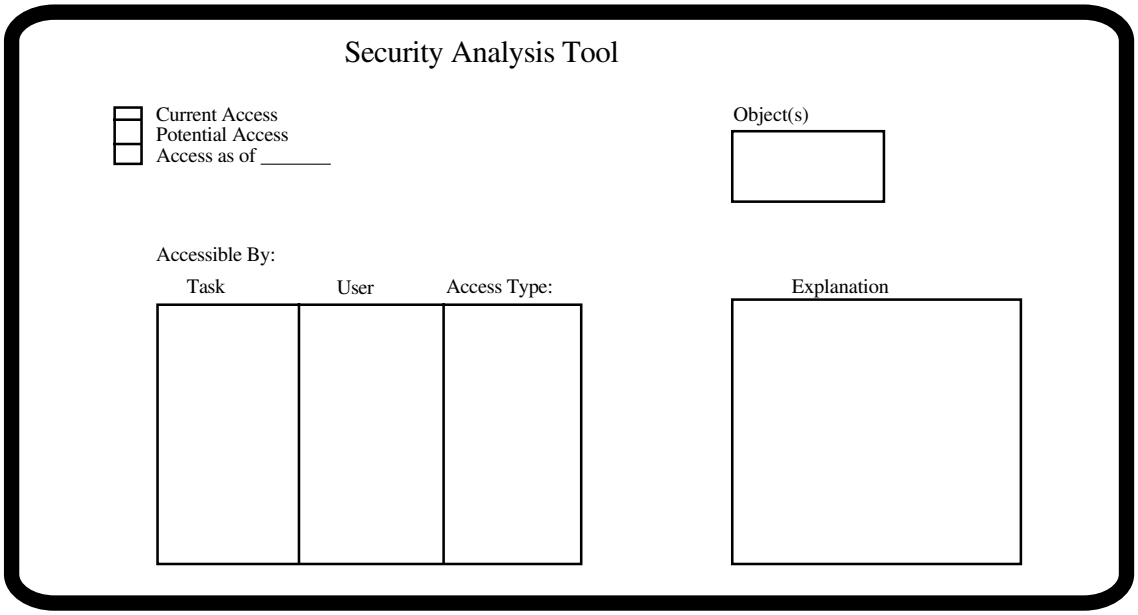

Fig. 6. Interface of Security Analysis Tool

Information can be requested according to the historical, current or future status of the knowledge base. Historical information requires the specification of a date and will check the access restrictions of the system as they existed at that date. Current status concerns the system as it exists now, while the future status provides information as the system may exist at a future point in time if all users and tasks receive access to all information they are potentially capable of receiving.

The user indicates which object is to be examined. If more than one object is specified then the security analysis tool reports on those users and tasks who have access to all of the specified objects. For example, a security administrator may want to know which users and tasks may access both, a patient´s insurance and a patient´s diagnosis.

The response of the security analysis tool is shown in the bottom half of the interface. A list of all tasks and users who may access the specified objects is returned along with the access types. Thereby the user knows who had, has, or is potentially able to have access to the specified objects. A user´s response could be to revoke access permission or to specify the denial of access to particular users or tasks.

Further information provided by the security tool enables a user to discover why access permission to an object is granted. The explanation component lists the objects in the knowledge base, including the rules, which result in a user and task receiving particular access permission.

4.2 Access and Potential Access to an Object

Determining who had access in the past, has access currently, or could have potential access to information in the future, is dependent on the security model which defines the capabilities of users in a system and the criteria necessary to access information. The GSM permits access to information if there is a link from the user´s task to the object. These links may be specified explicitly or implicitly by application of the rules in the GSM. Potential access is determined by checking which users and task possess, or may be granted access to an object.

Explicit Access. Explicit access means that an access type link exists from a task to the object. The security analysis tool grants the leader of the task which owns an object permission to read the users, their roles, as well as the tasks and the access types which are associated with the owner´s object. The owner of the object Bill may know all users and tasks who may access the object Bill.

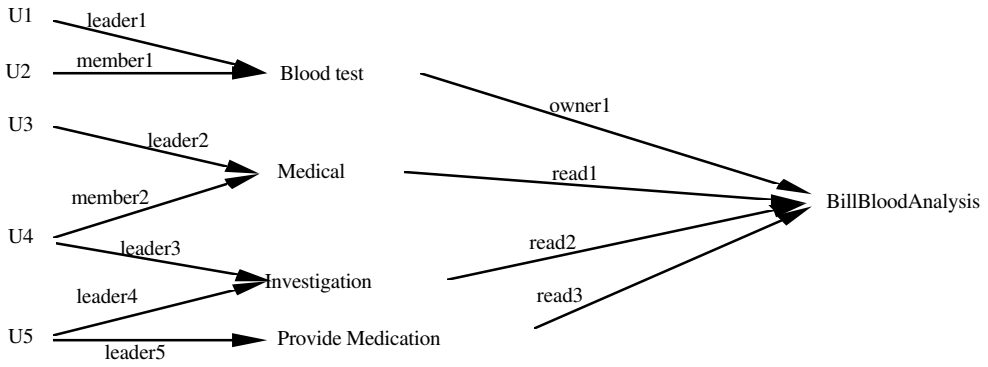

Fig. 7. Users and tasks with explicit access to an object.

Fig. 7 shows five users in four tasks with explicit access to the object BillBloodAnalysis. Since the Blood test task has owner access to the object, the leader of the Blood test task, U1, may know which other tasks and users may access BillBloodAnalysis, namely the users, their roles, their tasks and the access types which are shown.

In Telos these read access capabilities for the owner of an object are expressed as follows (notice that the information also requires that one is the leader of the task):

```
forall CU / currentuser, forall CT / currenttask,
forall X, ID1, ID2 / Proposition, forall T / tasks,
forall AT / tasks!accesstypes, forall R / users!roles,
forall U / users,

           A (CT, owner, X)
           and A (CU, leader, CT)
           and Prop (ID1, T, AT, X)
           and Prop (ID2, U, R, T)

                 ==>  A (CT, read, ID1 )
                      A (CT, read, T)
                      A (CT, read, ID2 )
                      A (CT, read, U)
```

The determination of who has access to deduced information is a more complex process. Deduced information is information that comes from the application of one or more rules. Deduced information does not have an owner. Even though deduced information is generated by an action of a user, it is not correct to speak of the user triggering a rule, as the owner of the information deduced from that rule. Ownership would include the capability of granting others access to the deduced object, as well as being able to delete the deduced object. These operations are not possible without modifying the rule from which the information was deduced.

Furthermore, the information generated by a rule depends on the knowledge base within which the rule is triggered. Therefore to determine who has access to some particular

deduced information would require being able to trigger the rules accessible to each user in the subset of the knowledge base which each user can access. It is therefore necessary to ask who has access to the rule which possibly leads to the deduced information, rather than to request who has access to some deduced information.

Implicit Access. Implicit access comes from the application of rules defined in the GSM. There are three areas where implied access types are generated: implied read rule, owner of class object, readquery access type.

The implied read access rule states that access to a link implies access to the source and destination of the link. For example, in fig. 7, if there was an attribute object Patient!test which has as destination the object BillBloodAnalysis, then all those who could read object Patient!test also have implicit read access to object BillBloodAnalysis.

The GSM also provides implicit access to tasks which are the owners of a class object. Owner access to a class object implies access permission to objects "above" and "below" the object with the owner access in order to enable full inheritance capabilities. The owner of the object BloodAnalysis of which BillBloodAnalysis is an instance, would have implicit access to the object BillBloodAnalysis.

A readquery provides a task with access to all objects which satisfy the query, with the query being evaluated when access is requested. To determine which users and tasks have access to an object, one must also determine if the object satisfies a query to which readquery access has been granted. If in fig. 7, the object BillBloodAnalysis satisfies a query object X, then all tasks with a readquery access type link to object X, have implicit access to object BillBloodAnalysis.

Potential Access. Reasoning about security allows one to determine all potential tasks which may receive access to an object, as well as all users who are, or may be associated with these tasks. In the following we assume that the user component of the GSM is fixed, i.e., no more users or supervise links are added to the knowledge base. In addition we assume that no further transfer links are created. By these assumptions, an analysis of potential access becomes possible through deductive database technology. If these assumptions are too limiting then further rules must be set up to handle additional potential users and tasks (and a careful meta level study of policy definitions will be needed for acyclicity and attenuation).

We first consider ***potential users*** for a task. Potential users are users who are, or may be, associated with a task. We use the term potential leaders since all potential users of a task, are potential leaders of that task.

The leader of a task may assign to his task or to his subtasks, as members or leaders, those users with whom he has a supervises relationship. The supervises relationship is

determined by a supervises link to another user or by the rule which states that the leader of a task supervises all users associated with the task.

The role potleader indicates a user who is a potential leader of a task and is expressed by the following two rules:

```
forall U1, U2 / users, forall T / tasks,

      ( A (U1, leader, T)
      or A (U1, potleader, T) )
      and A (U1, supervises, U2)
            ==>  A (U2, potleader, T)

forall U / users, forall T1, T2 / tasks,

      ( A (U, leader, T1)
      or A (U, potleader, T1) )
      and A (T2, partof, T1)
            ==>  A (U, potleader, T2)
```

Fig. 8 provides an example to show the range of potential users which can be assigned to a task. Currently only User1 and User2 have access objects associated with TaskA. As leader, User1 may assign additional users to TaskA, namely users whom he supervises, User10, User11, User100, User 101 and User110. If User1 should assign User2 also as leader of TaskA, then User2 would also have the capability to assign users to TaskA, namely User20, User21, User200, User210 and User211. Therefore all users shown are potential leaders of TaskA. All these users would also be potential leaders of a subtask of TaskA.

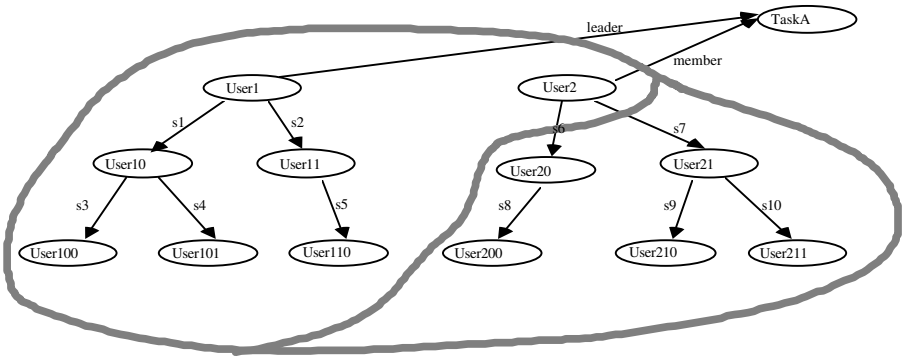

Fig. 8. Potential users who may be associated with a task.

Potential tasks w.r.t. an object, are tasks which may receive permission to access that object. A task receives access to an object in two ways. The first method is to be granted access permission from a task which owns the object. The second way that a task may receive access permisison to an object is if it is copied from the parent task.

A task with owner access type for an object may grant access permission to those tasks to which a transfer link exists. Therefore all tasks to which a transfer link exists are potential owners of the object. Each potential owner may in turn pass the object on to those tasks it has transfer links to.

The following rule determines for an object X, all those tasks which are the destination of a transfer link from an owner, and therefore are potential owners of the object X.

```
forall T1, T2 / tasks, forall X / Proposition,

            ( A (T1, owner, X)
            or A (T1, potowner, X) )
            and A (T1, transfer, T2)
                  ==>  A (T2, potowner, X)
```

Fig. 9 shows a task T1 which is the owner of an object X. Task T1 has a transfer link to task T2, and task T2 has a transfer link to task T3. Since task T1 is the owner of object X , and has a transfer link to task T2, task T1 may grant owner access of object X to task T2. The dotted line marked potowner1 indiates that task T2 is a potential owner of X. Should T2 receive owner access to object X, then it may grant owner access to task T3, since a transfer link exists from T2 to T3. T3 is thereby a potential owner of X.

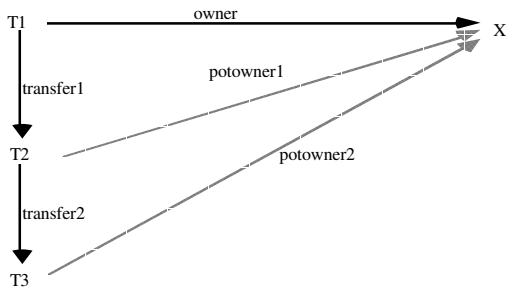

Fig. 9. Potential owners of an object

The second way a task may receive access permission to an object is from the parent task. In the GSM a parent task may copy the same access permission it has for an object, to its subtasks. Each subtask in turn, may copy the access permission it has to its subtasks. Therefore access permission to any task results in potentially all of its subtasks receiving the same access permission.

The fact that access permission associated with a task can be copied to its subtasks is expressed by the following rule:

```
forall T1, T2 / tasks, forall X / Proposition,
forall AT / accesstypes,

            A (T1, AT, X)
            and A (T2, partof, T1)
                  ==>  A (T2, potaccess, X)
```

We use `potaccess` to indicate any type of access. The actual potential access type is that possessed by the parent task.

In order to determine potential users and tasks we have defined users who may be associated with a task by the `potleader` role, and tasks which may be associated with an object with the `potowner` and `potaccess` access types. The owner of an object is granted access permission to all potential users and tasks of his object.

4.3 Explanation Component

The explanation component of the security analysis toolkit allows a user to determine how or why access to an object is granted. Access permission to an object may exist because of an explicit access type link between a task and the object. Access permission to an object may also exist from the application of rules which provide an implicit access type link between a task and an object. The explanation component displays the access type links and rules which result in the access permission.

The explanation component creates a query which searches the knowledge base for a particular task, access type and object. All rules and objects which contribute to the successsful discovery are provided in response to the query. The explanation component may also be used to explain how potential users and tasks may receive access to an object. This may be used in order that such potential access can be prevented before it exists.

5. CONCLUSION

Our goal in introducing task as a major concept in IS security modeling has been to make security impact analysis fit with the usual IS engineering process. A second potential benefit is a more fine-grained approach to the security analysis itself: there is an additional degree of freedom in defining user groups due to the assignment of task combinations to these groups.

The security toolkit, as implemented so far, has the advantage that no further software had to be written beyond the standard deductive database optimization techniques anyway present in our system. (There has been a substantial implementation effort, though, for providing an efficient way to store access rights, see [Steinke 92]).
On the other hand, we are quite aware that this paper is just a first step. In particular, we have mentioned the need for characterising GSM meta level policies in a way similar to those in SPM or related models. We do not see major problems as long as we follow simple policies such as providing each task leader full information about all the subtasks he creates (attenuation). However, it remains to be studied whether such a policy is

compatible with the delegation and privacy principles in organizations, and what we can say should that not be the case.

Literature

[Baldwin 87] Baldwin, R.W., Rule Based Analysis of Computer Security, In *Proc. of IEEE Compcon 87*, Feb. 87, 227-233.

[Berson & Lunt 87] Berson, T.A., Lunt, T.F., Multilevel Security for Knowledge- Based Systems, *Proc. of the 1988 IEEE Symposium on Security and Privacy*, 235-242.

[Borgida et al. 87] Borgida, A., Jarke, M., Mylopoulos, J., Schmidt, J.W., Vassiliou, Y., The Software Development Environment as a Knowledge Base Management System, In Schmidt/Thanos (eds.) *Foundations of Knowledge Base Management*. Springer-Verlag, 411-440.

[Bry et al. 88] Bry, F., Decker, H., Manthey, R., Integrity constraint satisfaction and satisfiability in deductive databases. *Proc. EDBT 88,* Venice.

[Garvey & Lunt 91] Garvey, T.D., Lunt, T.F., Multilevel Security for Knowledge Based Systems, Tech. Report SRI-CSL-91-01, SRI International, 1991.

[Glasgow et al. 91] Glasgow, J., MacEwen, G., Panangaden, P., A Logic for Reasoning about Security, unpublished paper, Aug. 7, 1991.

[Harrison et al. 76] Harrison, M.A., Ruzzo, W.L., Ullman, J.D., Protection in operating systems, *Communications of the ACM*, 19, 8, 1976, 461-471.

[Jarke 90] Jarke, M., DAIDA - Conceptual Modeling and Knowledge- based Support of Information Systems Development Processes, *Technique et Science Informatiques*, Vol. 9, No. 2, 121-133.

[Jeusfeld & Jarke 91] Jeusfeld, M., Jarke, M., From relational to object-oriented integrity simplification, *Proc. 2nd Intl. Conf. Deductive and Object-oriented Databases*, Munich 1991.

[Jeusfeld & Staudt 91] Jeusfeld, M., Staudt, M., Query optimization in deductive object bases. In Freytag/Maier/Vossen (eds.): *Query Processing in Object Databases,* Morgan Kaufmann, to appear.

[Mylopoulos et al. 90] Mylopoulos, J., Borgida, A., Jarke, M., Koubarakis, M., Telos: Representing Knowledge about Information Systems, *ACM Trans. on Information Systems*, Vol 8, No. 4, Oct. 90, 325-362.

]Rose et al. 92] Rose, T., Maltzahn, C., Jarke, M., Integrating object and agent worlds, *Proc. 4th Intl. Conf. Information Systems Engineering,* Manchester, UK, 1992.

[Sandhu 88] Sandhu, R.S, The Schematic Protection Model: Its Definition and Analysis for Acyclic Attenuating Schemes, *Journal of the ACM*, Vol. 35, No. 2, 1988, 404-432.

[Steinke 91] Steinke, G., Towards a Strategy for Achieving Security and Multi-User Integrity in Knowledge Base Systems, *Proc. 2nd Intl*

Workshop on the Deductive Approach to Informaiton Systems and Databases, Costa Brava, Catalonia, Spain, Sept 1991.

[Steinke 92] Steinke, G., Task-Based Security for Knowledge Base Systems, PhD Dissertation, University of Passau, 1992.