RTM*E*: Extension of Role-Task Modeling for the Purpose of Access Control Specification

Birgit Bomsdorf

University of Applied Science Fulda, Marquardstraße 35, 36039 Fulda, Germany
bomsdorf@hs-fulda.de

Abstract. Interactive systems are often developed without taking security concerns into account. We investigated a combination of both HCI models and access control specifications to overcome this problem. The motivation of a combined approach is to narrow the gap between different modeling perspectives and to provide a coherent mapping of modeling concepts. The general goal is a systematic introduction and tool support of security concerns in model-based development of interactive system. In this paper we report results of our work currently concentrating on the early design steps. The focus of this presentation is on the specification of task and role hierarchies, conflicting privileges and related tool support.

Keywords: Task modeling, Role modeling, Role task assignment, Tool support, Access control.

1 Introduction

Task and domain models are commonly used for the purpose of conceptual modeling. The combination of the two describes how users may manipulate objects while performing tasks. Access control management requires similar information, i.e. detailed specifications of the users' privileges to access objects and to perform operations on them. Security, however, is often postponed until the end of the design cycle or until the implementation of a system [4], [12]. Interactive systems are therefore developed without taking authorization concerns into account.

An access control model defines the permissions of users (e.g. human users, processes, computers) to access system resources (e.g. on an object, data base content, a file). Role is the central concept of prevalent access control models (Role-Based Access Control, RBAC [10]). Approaches such as [6], [8], [15] extending RBAC as well as the task-modeling approach TADEUS [13] have been showing the demand of differentiating between roles based on the structure of an organization (organizational roles) and roles based on the privilege to perform tasks (task-grouping roles).

In HCI this distinction exists but is hardly introduced into the kernel concepts of tools supporting task-based modeling. In some tools an explicit role model does not exist (K-MADe [1], Diane+/Tamot [7], TaskArchitect [14]). In contrast to this, in CTTE [9] separate task models are created per role defining all tasks that can be performed by that role. The role specification, however, does not support inheritance of

R. Bernhaupt et al. (Eds.): HCSE 2010, LNCS 6409, pp. 149–157, 2010.

privileges. This is supported in WSDM [3] but resulting role hierarchies are not formally integrated with task models. All in all, concepts such as agent, actor, role, and group are used equivocally and ambivalently even within a single HCI approach. Furthermore, (semi-) automatic support in modeling the group, role, and task hierarchies taking care of their mutual dependencies is hardly supported. These facts complicate the combination of HCI modeling with security concerns.

Guo [5] has been showing in his work how the complexity of the three hierarchies and their relationships may be handled in the context of access rights. Based on his proposal we extended our modeling approach [2]. The extensions are also combined with MAD [1] aiming at modeling extensions in general. Therefore, for the remaining of the paper we refer to it by the abbreviation RTM*E* (Role-Task Model *E*xtension). In the following, as indicated by the name, the focus is on tasks, roles and their mutual dependencies. The approaches reported in [12] and in [4] are comparable with our work. They aim at the integration of access control specifications with models known from Software Engineering and Web-Engineering, respectively. However, privileges are formulated by means of roles and system functionality, not considering the context of users performing tasks to reach goals.

2 RTM*E*: Integrative Modeling

Fig. 1 depicts an overview of the model and the interrelations of our integrative approach. It enables the explicit specification of organizational roles, named *groups*, as well as of task-oriented *roles* to define privileges. Users (individual persons) are specified as well. They are members of groups whereby, because of *adopts* relations, they are enabled to perform tasks (*performs* relation) and hence to act on objects (*involves* relation), i.e. to invoke methods. Our current tool, which is based on these concepts, supports the creation of group, role, and task hierarchies taking into account their interdependencies. Underlying functions check consistency violations and support to solve them. Hereby they contribute to the reduction of modeling complexity and to avoid modeling errors.

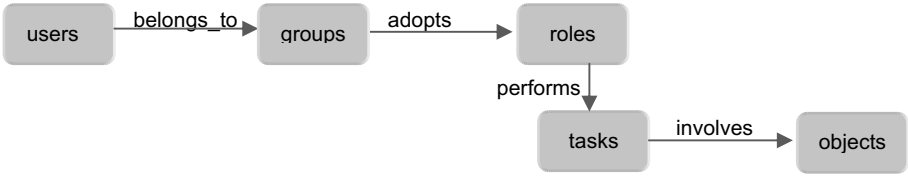

Fig. 1. Overview of modeling concept

2.1 Task Model

Figure 2 depicts a task model example by means of the notation as introduced in MAD [1]. The simplified task model consists of three unconnected task trees. The task *online shopping* is a composition of the subtasks *rate seller* and *buy product*, which is further refined. The task *sell online* is subdivided into two subtasks, whereas *administrate website* simply consists of a single task. Additional concepts are commonly in use to

specify the order of task execution, such as temporal relations, and conditions constraining task performance, e.g. pre- and post-conditions. These concepts are not detailed here since the subsequent considerations reference only the task hierarchy. The following remarks are added only to complete the description of the example: The sequencing of the direct subtasks of *online shopping* as well as of *sell online* is open because of the *No order* declaration in each case. The option *enabling* defines that subtasks are to be performed one after the other, in which the sequence is given by the graphical order from left to right in the diagram. *Elementary* is used for leaf tasks, i.e. for tasks without subtasks. The label *OPT* denotes optional task execution.

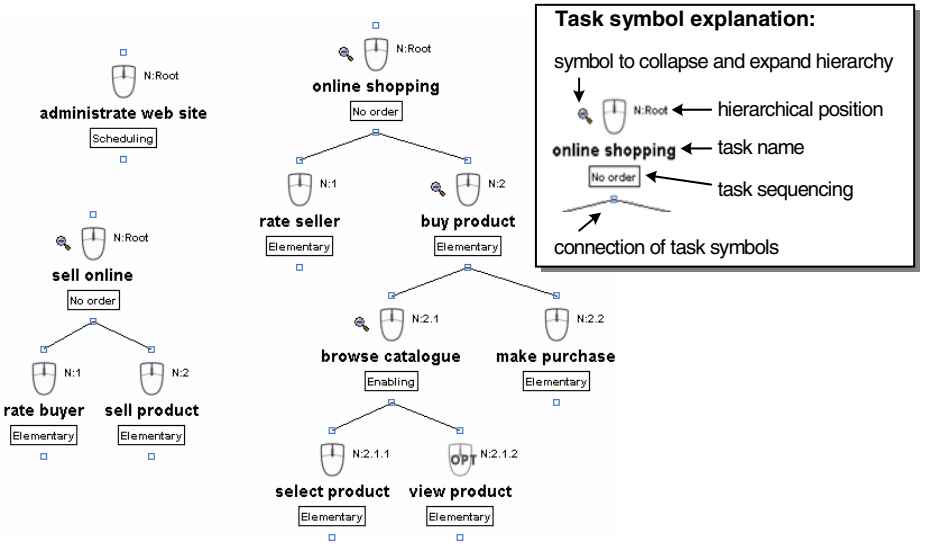

Fig. 2. Example of a task and a role model

2.2 Role Model

A role-task-mapping "r *performs* t" (see figure 1) means that the user who has taken role r possesses the privilege to perform t. The set of all privileges of a role r is denoted by *privileges*(r). Roles are structured by a so called poly-hierarchy that is given by a graph. While the task hierarchy expresses composition relations, the role hierarchy describes inheritance of privileges. An edge from a role r1 to a role r2 indicates that the privileges assigned to role r1 are a proper subset of the privileges of r2, i.e. $privileges(r1) \subset privileges(r2)$. For this kind of relation it is said that r1 is junior to r2 and that r2 is senior to r1 [10].

In our example we want to allow each person to browse the catalogue. However, only persons adopting a role *buyer* should be enabled to buy a product and to rate a seller. First of all we create a role *everyone* and assign to it the task *browse catalogue*. Fig. 3 left hand shows the result of this editing step. Inserting the first role is done easily. It is positioned between *minRole* and *maxRole* as shown in the example with $privileges(minRole) \subset privileges(everyone) \subset privileges(maxRole)$.

Each graph posses a minimal and a maximal role that are introduced for the purpose of computing a role model's hierarchy and taking care of conflicts. A detailed description is provided in [5]. Please note that the set of privileges of *minRole* is always empty while *maxRole* possesses all the time all privileges defined by the task model.

Now we create a role *buyer* and assign the task *online shopping* and hereby also all of its subtasks to the role (see *Inherited role task(s)* in the role editor window). This step results in a model (see Fig. 3 bottom), in which *buyer* is positioned into the role hierarchy according to the privileges added to the role as well as to the hierarchy existing so far. The task *browse catalogue* is part of *online shopping*, i.e. *privileges*(*everyone*) ⊂ *privileges*(*buyer*) holds true, based on which the position is determined. Similarly the roles *seller* and *administrator* are inserted to define privileges for *administrate web site* and *sell online*. These two tasks are not connected to the tasks taken into account so far. Hence, both roles are inserted separately into the role hierarchy (see figure 5 right hand). Explanations of more complex cases can be found in [5] and [11].

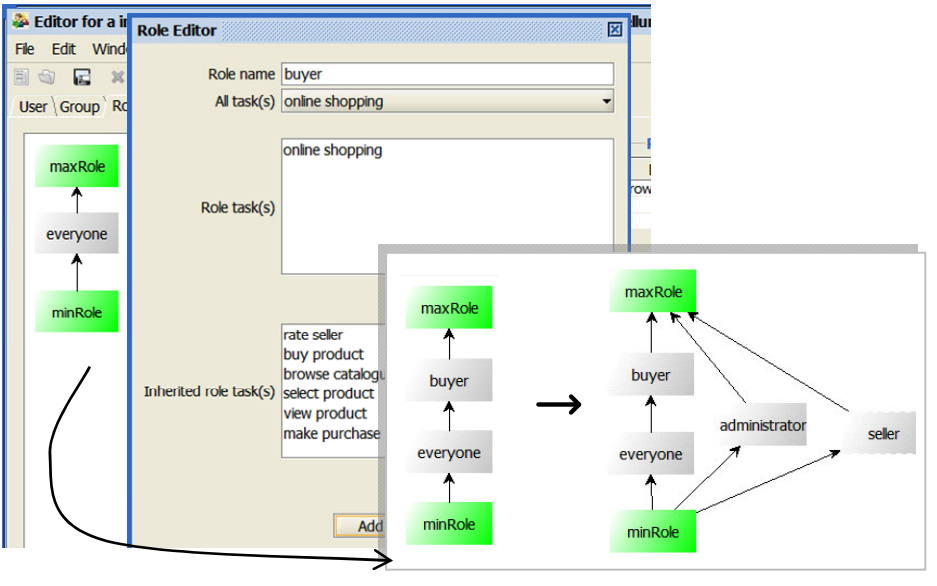

Fig. 3. Inserting the role *buyer*

2.3 Conflicts

Task models as well as role models mostly lead to complex hierarchical structures. Complexity is increased by their mutual dependencies. Thus, different conflicts within a specification may result from assigning roles and tasks to each other. The existing hierarchical structures of both the tasks and the roles are to be considered. It is not allowed to assign more tasks to a junior role than to one of its senior roles. This would result in an *assignment conflict* because in such a case *privileges*(*senior role*) ⊂

privileges(*junior role*) would hold true. This, however, is conflicting with the definition of the role hierarchy (given above). A strategy for avoiding such modeling errors is incorporated in RTM*E*. Each time a user of our role editor selects a task that would cause a conflict a warning is shown and the user is prompted to perform a correction. Similarly RTM*E* evaluates editing steps while the task model is under construction or is being modified.

In addition to such *assignment conflicts* RTM*E* enables to define explicit cases of conflicts, named *privilege conflicts* and *role conflicts*, which are checked during modeling.

2.3.1 Role Conflicts

A *role conflict* is defined for two roles r1 and r2, formally denoted by r1 ←→ r2. It specifies that roles are mutually exclusive. In such a case a user must not take both roles; it is also forbidden to assign the two roles to the same group. Hence, either r1 can be assigned to a group g or r2 can be assigned to g. Furthermore, for r1 ←→ r2 it must hold true that no path exists between r1 and r2 in the role graph.

Basically, the two roles r1 and r2 must be independent from each other, i.e. the only common junior is *minRole* [5]. (A role is a common junior role of two roles if it is junior of both.) In addition, *maxRole* must be the only senior role of the two conflicting roles r1 and r2 [5]. Thus, a role conflict can only exist if r1 is neither senior nor junior to r2. Otherwise, r1 and r2 would have at least a common privilege violating the mutual exclusiveness.

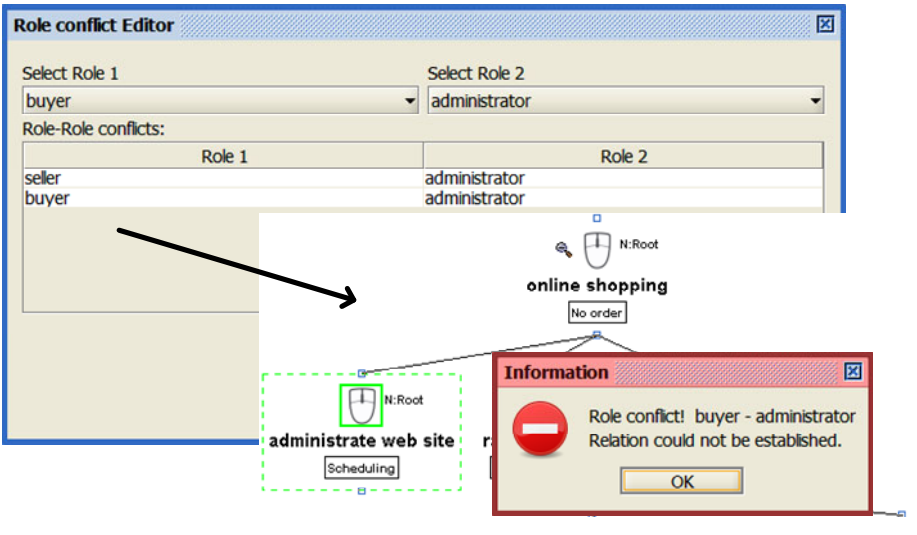

Fig. 4. Role conflict example

Obviously, a defined role conflict does not only impact group but also task modeling. Fig. 4 shows two role conflicts specified for the example given above: *seller* ← → *administrator* and *buyer* ← → *administrator*. Hence, common privileges of the roles *seller* and *administrator* as well as of the roles *buyer* and *administrator* are

mutually exclusive. The attempt, for example, to define the task *administrate web site* as a subtask of *online shopping* results in an error now. Please note that the task *online shopping* is assigned to the role *buyer* while *administrate web site* is assigned to *administrator*. In the case *administrate web site* should become a subtask of *online shopping*, the role model would have to be modified as well so that a junior relation exists between *buyer* and *administrator*.

2.3.2 Privilege Conflicts

In addition to formulating conflicts between roles RTM*E* enables to define conflicts of privileges. A *privilege conflict* specifies two tasks t1 and t2, noted by t1 < > t2, that must not be assigned to a role r at the same time. The role *maxRole* is an exception to this rule as it comprises all privileges existing in the model. However, this exception causes no problem since *maxRole* cannot be assigned to any user or group of users.

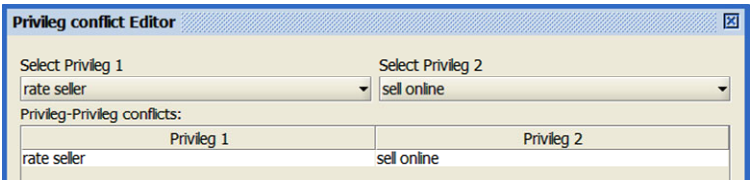

Fig. 5. Specification of a privilege conflict

Fig. 5 shows a privilege conflict defined for the tasks *rate seller* and *sell online* (*rate seller* < > *sell online*). Hence, *rate seller* and *sell online* must not be assigned to the same role. The role editor in Fig. 6 shows the attempt to allow a user who takes the role *seller* to rate sellers. However, the task *sell online* has been assigned to *seller* and thus the set *privileges*(*seller*) cannot be expanded by *rate seller*. RTM*E* forbids the modification showing an error message. It is up to the developers to decide on corrections. For example, they might remove the privilege conflict (*rate seller* < > *sell online*). In the same way *privileges*(*buyer*), that already contains *rate seller*, cannot be extended with *sell online* (see for privileges of buyers the role editor content shown in Fig. 3).

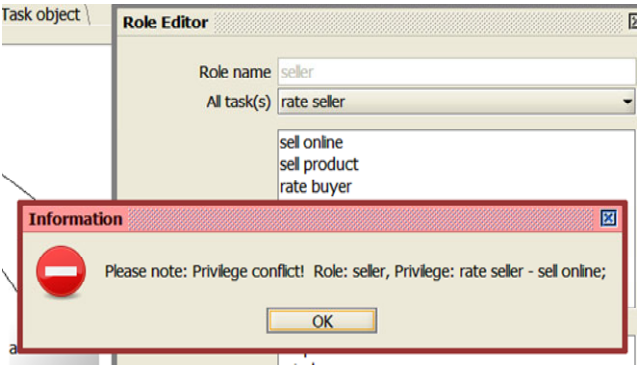

Fig. 6. Privilege conflict impacting role modeling

The defined privilege conflict has a similar impact on task modeling. The task *sell online* cannot be specified as a subtasks of *rate seller* (see Fig. 7). If the *rate seller-sell online* conflict is not specified the role seller would become a junior role of *buyer* because of the automatic role structure computation.

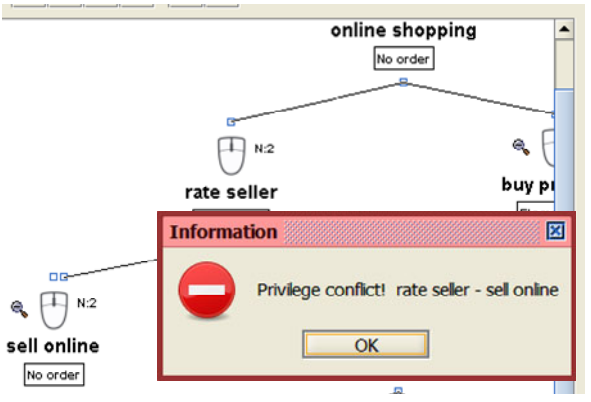

Fig. 7. Privilege conflict impacting task modeling

3 Conclusion

The modeling steps presented in this paper basically consist of privilege definitions. Hereby the groups a person belongs to, the roles a person may adopt, the tasks a person can perform, and the objects a person is allowed to access are specified. The definition of group structures, similarly to role modeling as presented above, is done by subset relations (also resulting in an acyclic graph with directed edge). RTM*E* handles the specification of group hierarchies and dependencies on role models in a very similar way as role-task modeling.

The underlying theory combines our own work on task modeling [2] with the work on conflict handling by [5]. Hence, our approach provides not only a sound theoretical basis but contributes also to integration of HCI with Access Control. The extensions with respect to our previous work are twofold: On the one hand the differentiation of organizational groups and roles defining privileges is incorporated. On the other hand RTM*E* implements an integrative group, role and task modeling approach. RTM*E* assists in structuring the model taking into account model properties defined so far. Each time a modeler inserts an additional assignment the resulting hierarchies are determined and the new assignment is inserted only if it causes no conflicts. This technique allows, for example, assigning roles and tasks to each other without performing required restructuring of the hierarchies. RTM*E* evaluates instead the new graph models considering resulting and explicitly defined rules of conflicts. This approach meets the fact that the model hierarchies result from the groupings of privileges.

Construction of model hierarchies and conflict control is based on the rules defined for the meta-model (assignment conflicts) together with the conflict rules defined by the modeler, namely role conflicts and privilege conflicts.

In the examples modeled so far by means of RTM*E* the underlying algorithms have proven to be very useful. However, the models were relatively small. We are aware that in the context of real projects more support is needed to reduce complexity and mental load. A common technique is to provide various views on a model in conjunction with filtering mechanisms. The first implementation of RTM*E* [11] provides such view generation. The modeler can, for example, extract all objects a special person has access to or all persons that are allowed to take a specific role. The interactive extracted views are presented by a diagram similarly to those used during editing.

The RTM*E* editor is in the state of a prototype. It possesses import and export functions (on the basis of XML) enabling the exchange of models with other tools. Currently our own task modeling approach and MAD are supported. Generally, RTM*E* can be combined with task models in which a super task is the sum of its subtasks, i.e. the superior task does not define additional functionality.

Acknowledgments. The author likes to thank Andreas Reitschuster for his contribution to this work.

References

1. Baron, M., Scapin, D.: K-MADe User Manual (2006), `http://kmade.sourceforge.net` (accessed 22.8.2010)
2. Bomsdorf, B.: The WebTaskModel Approach to Web Process Modelling. In: Winckler, M., Johnson, H., Palanque, P. (eds.) TAMODIA 2007. LNCS, vol. 4849, pp. 240–253. Springer, Heidelberg (2007)
3. Casteleyn, S., De Troyer, O.: Structuring Web Sites Using Audience Class Hierarchies. In: Arisawa, H., Kambayashi, Y., Kumar, V., Mayr, H.C., Hunt, I. (eds.) ER Workshops 2001. LNCS, vol. 2465, pp. 198–211. Springer, Heidelberg (2002)
4. Díaz, P., Aedo, I., Sanz, D., Malizia, A.: A model-driven approach for the visual specification of Role-Based Access Control policies in web systems. In: Visual Languages and Human-Centric Computing, VL/HCC 2008, pp. 203–210 (2008)
5. Guo, Y.: User/Group Administration for RBAC. The University of Western Ontario (1999)
6. Kang, M.H., Parker, J., Forscher, J.N.: Access Control Mechanisms for Inter-Organizational Workflow. In: Symposium on Access Control Models and Technologies, pp. 66–74 (2001)
7. Lu, S., Paris, C., Vander Linden, K.: Tamot: Towards a Flexible Task Modeling Tool. In: Proceedings of Human Factors, pp. 878–886 (2002)
8. Osborn, S., Nyanchama, M.: The Role Graph Model and Conflict of Interest. The University of Western Ontario (1999)
9. Paternò, F., Santoro, C., Tahmassebi, S.: Formal Models for Cooperative Tasks: Concepts and an Application for En-Route Air Traffic Control. In: Design, Specification, and Verification of Interactive Systems, pp. 71–86. Springer, Abingdon (1998)
10. RBAC Standard (2003), `http://csrc.nist.gov/rbac/rbac-std-ncits.pdf` (accessed 22.8.2010)
11. Reitschuster, A.: Realization of an Editor supporting an integrative Modeling of Groups, Roles and Tasks (Realisierung eines Editors zur integrativen Gruppen-, Rollen- und Aufgabenmodellierung, in German), Masterthesis, University Hagen (2008)

12. Romuald, T., Stéphane, C.: Integration of Access Control in Information Systems: From Role Engineering to Implementation. Informatica 30, 87–95 (2004)
13. Stary, C.: Role-Adapted Access to Medical Data: Experiences with Model-Based Development. In: Universal Access in Health Telematics, pp. 224–239 (2005)
14. Stuart, J., Penn, R.: TaskArchitect: taking the work out of task analysis. In: 3rd Annual Conference on Task Models and Diagrams, TAMODIA 2004, pp. 145–154 (2004)
15. Zhang, C., Hu, Y., Zahng, G.: Task-Role Based Dual System Access Control Model. International Journal of Computer Science an Network Scurity 6(7B), 211–215 (2006)