Specifying Separation of Duty Constraints in BPEL4People Processes

Jan Mendling[1], Karsten Ploesser[2], and Mark Strembeck[3]

[1] BPM Cluster, Faculty of IT, Queensland University of Technology
126 Margaret Street, Brisbane QLD 4000, Australia
j.mendling@qut.edu.au
[2] SAP Research, Brisbane QLD 4000, Australia
karsten.ploesser@sap.com
[3] Vienna University of Economics and Business Administration
Institute of Information Systems, New Media Lab, Austria
mark.strembeck@wu-wien.ac.at

Abstract. Security issues have to be carefully considered for information systems that support the business processes of an organization, in particular, when these systems build on open interfaces such as web services. In this paper, we examine the new BPEL extension *BPEL4People* from an access control perspective. In particular, we discuss the importance of "separation of duty" constraints and identify options to specify such constraints in BPEL4People processes. Moreover, we identify and discuss shortcomings of the BPEL4People specifications that complicate and/or impede separation of duty enforcement. In addition, we suggest solutions which can be introduced into future versions of BPEL4People to mitigate those shortcomings.

1 Introduction

The standardization of business process management and workflow technology has been discussed for more than ten years, and several standardization bodies have proposed specifications for different aspects of business process management (see, e.g., [16]). Since 2003, the Organization for the Advancement of Structured Information Standards (OASIS) has driven the specification of the Business Process Execution Language for Web Services (BPEL) [4] which has become an important standard in this area thanks to its extensive support by major software vendors.

Originally, the BPEL specification was lacking a generic concept for activities that are performed by human agents. Due to this missing feature, BPEL could hardly be used as a platform-independent format for describing and exchanging human workflows. For this reason, major software vendors have been working on the specification of human activities in BPEL based on its extension mechanism. As a result, two complementary draft specifications were recently released: BPEL4People [2] and WS-HumanTask [1]. WS-HumanTask defines the lifecycle and the generic roles associated with a particular task, and BPEL4People defines

how such tasks can be integrated in a BPEL process. In this paper, we refer to both documents together as the BPEL4People specifications, or just B4P/HT.

While the introduction of B4P/HT is a valuable extension to BPEL, it raises several questions from an access control perspective. When performing an IT-supported workflow, human users and proactive/autonomous software agents have to fulfill certain tasks to process the workflow. Each action in a workflow (like changing a document or sending a message) is typically associated with a certain access operation (e.g. to a document or a messaging service). Thus, an active entity participating in a workflow (be it a human user or a software agent) must be authorized to perform the actions that are needed to complete its tasks. The business processes of an organization are therefore an ideal source to define a tailored set of *access control policies* (also referred to as *authorization policies*) for this organization, respectively their information system (see [18, 25]).

Access control deals with the elicitation, specification, maintenance, and enforcement of authorization policies in software-based systems [15, 22]. In role-based access control (RBAC) [10], roles are used to model different job-positions and scopes of duty within a particular organization and/or within an information system. These roles are equipped with the permissions that are needed to perform their respective tasks. Human users and/or other active entities (subjects) are assigned to roles according to their duties, respectively their work profile (see [18, 24]). The descriptions of roles tend to change significantly slower than the assignment of individuals to these roles. Thus, establishing roles as an abstraction mechanism for subjects facilitates the administration of the access control policies. Moreover, the advantages of RBAC on the modeling and technical level directly translate into lower maintenance costs [11].

A particular access control specification is said to be *safe* iff no subject can obtain an "unauthorized" right. However, since the verification of the safety property for general access control models like RBAC is not decidable [13], *constraints* are often used to enforce the safety property via explicit modeling-level artifacts. *Separation of duty constraints* enforce conflict of interest policies (see, e.g., [9, 23]). Conflict of interest arises as a result of the simultaneous assignment of two mutual exclusive permissions or roles to the same subject. *Mutual exclusive* roles, or permissions result from the division of powerful rights or responsibilities to prevent fraud and abuse. An example is the common practice to separate the "controller" role and the "chief buyer" role in medium-sized and large companies. Two mutual exclusive roles are not allowed to be assigned to the same subject, and two mutual exclusive permissions must not be assigned to the same role or the same subject. Moreover, in workflow environments it is of central significance that separation of duty constraints can also be defined and enforced on the level of tasks (see, e.g., [7]).

There has been some work on access control for BPEL processes (see, e.g., [5, 14, 17]). However, an analysis of the B4P/HT specification from a separation of duty perspective is missing so far. In this paper, we thus address the enforceability of separation of duty constraints in B4P/HT. This includes questions like: How can we express in B4P/HT that a certain individual is not allowed to do

both the "check loan application" task and the "decide about loan acceptance" task for one particular loan application? And if B4P/HT provides means to express such separation of duty constraints, are there ways to circumvent them?

The remainder of this paper is structured as follows. In Section 2 we introduce the concepts of B4P/HT which are important for the purposes of this paper. Section 3 then gives an overview of role-based and task based access control, and discusses the relevance of separation of duty constraints. Subsequently, Section 4 provides an analysis of B4P/HT from the separation of duty perspective. In particular, we discuss how specific aspects of separation of duty constraints can be enforced in B4P/HT, identify corresponding shortcomings of the B4P/HT specifications, and suggest solutions which can be introduced into future versions to mitigate those shortcomings. Next, Section 5 discusses related work before Section 6 concludes the paper.

2 Preliminaries on B4P and WS-HT

In this section we give an overview of BPEL4People and WS-HumanTask (B4P/HT). For this introduction there is no detailed knowledge of BPEL required. In essence, a BPEL Process defines a set of activities and their control flow. BPEL 2.0 [4] offers a so-called "extension activity" which is used by B4P/HT for adding human tasks to BPEL. B4P/HT defines three generic process roles: the process initiator, process stakeholder, and business administrator. While in most cases the process initiator will be determined by the BPEL engine at runtime, the other two generic roles are populated by evaluating a people query. We use the term "people query" to abstract from the three different options provided by B4P/HT of assigning people to roles, i.e. using the so-called logical people group which is an alias to a query, using literals, or using expressions (see [1, 2]). The specification, however, does not mandate a concrete people query language, which leaves room for vendor-specific implementations. The BPEL process related concepts of B4P/HT are depicted on the left-hand side of Figure 1.

The right-hand side of this figure shows the task concept, its generic roles, the task lifecycle states, and lifecycle transitions. These concepts are defined in the WS-HumanTask specification. A single task has different generic roles assigned to it. Similar to the overall process there are task initiators, task stakeholders and business administrators. Beyond that, a task has a set of potential owners, one actual owner, and a list of excluded owners. These three generic roles define in essence who *may* legally be assigned to a task and who *is* actually assigned to a task. The people that populate these different task generic roles can influence the lifecycle of a task at predefined stages.

The typical sequence of B4P/HT task states is from *created* to *ready* to *reserved* to *in-progress* to *completed* [1, p.37]. Depending on the task state and on their assignment to one of the generic task roles, people can control the task lifecycle via predefined transitions. Several of these transitions are only allowed for members of the *potential owner* role, including to claim and to start, to

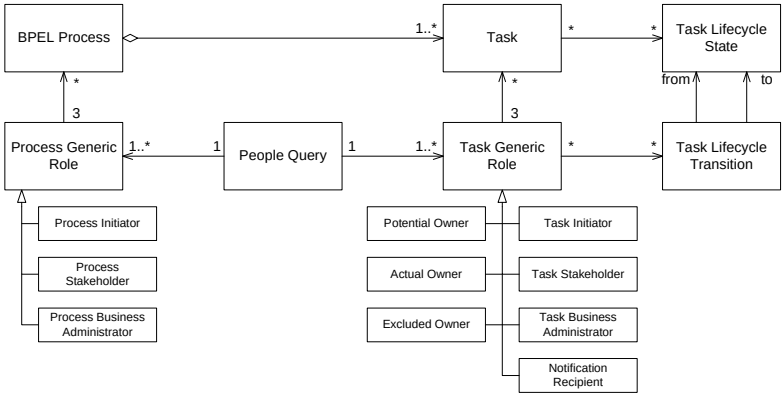

Fig. 1. Class diagram: BPEL4People

suspend and to resume, as well as to delegate and to forward a task. Both the latter (forward and delegate) are interesting from a security perspective. When a task is delegated, the delegatee becomes the actual owner of the task and is added to the potential owners list if she was not before (see also Section 4). The set of potential delegatees can be limited for each task to "anybody", "nobody", "potential owners", or to a list of predefined people ("others"). The forwarding mechanism works similar with the only difference that it cannot be restricted and that the forwardee replaces the forwarder in the potential owners list. Most of these transitions are also allowed for members of the *actual owner* role. Beyond that, an actual owner can stop, release, complete, signal failure, and skip a task.

The *business administrator* is the most powerful role since all state transitions are accessible to it. The business administrator is important if the list of potential owners is evaluated to the empty set. In this case, he nominates one person to execute the task. Beyond that, B4P/HT allows to define that certain people may not perform a particular task via the generic role *excluded owners*. Users being assigned to the "excluded owners" role are thus excluded from the potential owners list, and cannot progress the task in its lifecycle.

Furthermore, B4P/HT standardizes methods to access the assignment history of a particular workflow case. These methods can be used to query the members of a generic role for a particular task. For instance, the *getActualOwner(X)* function returns the actual owner of task X. This way, one can retrieve role members for previously executed tasks, for instance, to exclude them from other tasks. We will elaborate on this feature later when we discuss how B4P/HT can be tailored to support task-based separation of duty constraints.

3 Role-Based and Task-Based Access Control

In order to allow for an (automated) enforcement of authorization policies, the high-level control objectives specified for a system need to be mapped to the

structures provided by an access control model. An *access control model* provides an abstract framework for the definition of authorization policy rules. It defines how essential access control elements (like subjects, operations, objects) could be interrelated. In addition, it may specify invariants which must be met by each real-world implementation of this model (e.g. to enable control of information flows) or predetermine administrative procedures.

In recent years, role-based access control (RBAC) [10] – together with diverse extensions and variants – has evolved into a de facto standard for access control in both research and industry. One of the advantages of RBAC is being a general access control model. This means that a sophisticated RBAC-service may be configured to emulate many different access control models, including discretionary and mandatory access control models (see [19]).

A central idea in RBAC is to support constraints on almost all parts of an RBAC model (e.g. permissions, roles, or assignment relations) to achieve high flexibility. Static and dynamic separation of duty constraints (see [8]) are two of the most common types of RBAC constraints (see, e.g., [3]). Separation of duty (SOD) constraints can be subdivided in static separation of duty (SSD) constraints and dynamic separation of duty (DSD) constraints. *Static separation of duty* constraints specify that two mutual exclusive roles (or permissions) must *never* be *assigned* to the same subject *simultaneously*. *Dynamic separation of duties* constraints define that two mutual exclusive roles (or permissions) must *never* be *activated* by the same subject *simultaneously*. This means that two dynamically mutual exclusive roles may be assigned to the same subject. The corresponding subject, however, is only allowed to activate at most one of its dynamically mutual exclusive roles (permissions) at the same time.

Various contributions concerning access control in collaborative environments exist, esp. for groupware and workflow systems. For example, Thomas and Sandhu introduced TBAC [29], a family of models that support the specification of active security models. In TBAC, permissions are actively (de)activated according to the current task/process-state. In [6], Bertino et al. present a well-elaborated language and algorithms to express and enforce constraints which ensure that all tasks within a workflow are performed by predefined users/roles. In [12], Georgiadis et al. introduce the Context-based Team Access Control model (C-TMAC) as an extension of the TMAC approach presented by Thomas [28]. Here, a team is defined as a group of users acting in different roles with the objective of corporately completing a certain task, e.g. a group of physicians and nurses attending a patient. Thus, in C-TMAC the team concept is used to associate users with contexts, like roles are used to associate users with permissions. One similarity for all of these approaches is that they facilitate the usage of some context information, e.g. the execution history of individuals/roles and the current task, to make assignment, activation, or authorization decisions.

In this paper, we are especially interested in the specification and enforcement of task-based separation of duty constraints in the B4P/HT context. Here, a *task-based separation of duty constraint* is a separation of duty constraint that considers task order and task history in a particular process instance to decide

if a certain subject or role is allowed to perform a certain task (see also [7, 30]). Again, task-based SOD constraints can be static or dynamic. A *static task-based SOD constraint* defines that two statically mutual exclusive tasks must never be assigned to the same role and must never be performed by the same subject. This constraint is global with respect to all process instances in the corresponding information system. For example, a company may choose to define two tasks "Order supplies" and "Approve payment" as statically mutual exclusive to prevent fraud and abuse.

In contrast, a *dynamic task-based SOD constraint* refers to individual process instances and defines that two dynamically mutual exclusive tasks must never be performed by the same subject in the same process instance. In other words: two dynamically mutual exclusive tasks can be assigned to the same role. However, to complete a process instance which includes two mutual exclusive tasks, one needs at least two different subjects (i.e. two individuals owning the respective role). This means, although a subject might possess a role which includes all permissions to perform two dynamically mutual exclusive tasks, a dynamic task-based SOD constraint can enforce that the same subject does not perform both tasks in the same process instance. For example, a bank may assign two tasks "Check credit worthiness" and "Approve credit application" to the "Bank clerk" role and define a dynamic task-based SOD constraint on these tasks. Each subject owning the bank clerk role may then perform both tasks. Nevertheless, because of the dynamic SOD constraint on these tasks, we always need at least two bank clerks to complete a "Credit application" process.

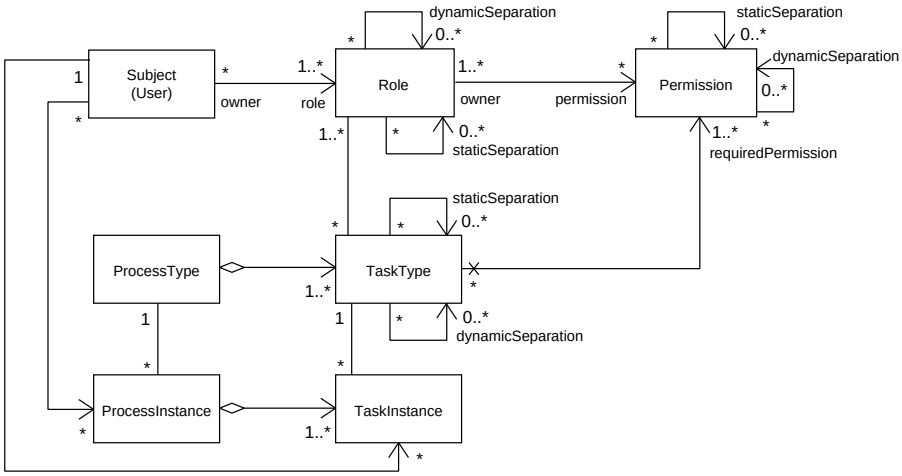

Fig. 2. Class diagram: task and role-based access control

Figure 2 shows the essential relationships of subjects (users), roles, permissions, tasks, and processes that are important for the purposes of this paper.

In general, static as well as dynamic SOD constraints can be defined on the level of roles, permissions, and tasks – resulting in a total of six different types of SOD constraints. Moreover, SOD constraints are subject to inheritance (see, e.g., [9, 23]). For example, if one of two statically mutual exclusive tasks is assigned to a certain role, the other task must be assigned to an other role. Subsequently, the corresponding roles are statically mutual exclusive because they inherit the corresponding SOD constraints from their assigned tasks. Moreover, two tasks are mutual exclusive if one needs two mutual exclusive permissions to complete both tasks. In this case, the SOD constraint between two tasks is inherited from the permissions that are needed to perform these tasks (see also Figure 2).

4 BPEL4People Support for SOD Constraints

In this section, we discuss in how far the six types of separation of duty constraints (see Section 3) can be implemented in B4P/HT. First, Section 4.1 presents strategies for adding SOD support to a B4P/HT process. Subsequently, Section 4.2 discusses access control enforcement issues in B4P/HT and proposes solutions.

4.1 Strategies for Implementing SOD in B4P/HT

WS-B4P/HT does not advocate a specific mechanism to implement access control. More specifically, the specification states that a "mechanism determining how an implementation evaluates people assignments" is out of scope. Given this design choice there are basically two ways to implement access control within B4P/HT: firstly, via the enforcement of access control policies in people queries, and secondly, by enforcing access control policies in the people assignment section of the respective tasks. At the time of writing, there is no standardized people query language available to be used with WS-B4P/HT. Accordingly, no assumptions can be made about access control mechanisms in such a language. As a consequence of that, we focus on the second approach[1].

Role-based constraints play an important part in the specification of access control. However, even though B4P/HT utilizes people queries as a role-like concept, it does not provide means to define direct relations between roles such as inheritance relations or separation of duty relations. More precisely, B4P/HT states that "the structure of the data in the people directory is [...] out of scope" (see [1, 2]). Below, we now discuss if and how the different types of separation of duty constraints can be supported based on the options provided through the current B4P/HT specifications:

[1] Note that a third option is the extension of B4P/HT with explicit (implementation independent) concepts to enable the direct and explicit integration of access control relevant information in B4P/HT task definitions. Our analysis provides the foundation for considering suitable extensions.

Permission-based SOD: Since B4P/HT does not support a notion of permission, corresponding SOD constraints need to be captured in people queries or in the organization model outside B4P/HT.

Static role-based SOD: In essence, static role-based separation of duty demands that two mutually-exclusive roles must not be assigned to the same subject. This requirement can be partially translated into the people query concept of B4P/HT. When two people queries pq_1 and pq_2 should be mutually exclusive, this can be enforced via the generic role assignment of a task in the following way: tasks whose potential owners are populated with pq_1 get pq_2 assigned to the excluded owners (see Section 2). As a consequence, there will be no task in the process for which pq_1 and pq_2 can yield the same subject.

Dynamic role-based SOD: Dynamic role-based SOD demands that two mutual exclusive roles can never be activated by the same subject. We again consider two people queries pq_1 and pq_2 for this case. This requirement can be enforced by adding each actual owner ao_1 and ao_2 of all tasks that use pq_1 and pq_2 to get their potential owners as excluded owners. This means, if task t_1 has the extension of pq_1 as its potential owners, then it must exclude all actual owners of tasks that have pq_2 as potential owners. The actual owners can be retrieved using the getActualOwner method of B4P/HT.

Static task-based SOD: Static task-based separation of duty demands that two mutual exclusive tasks must not be assigned to the same role. While B4P/HT does not directly offer a user-configurable role concept, this constraint can be enforced using two people queries pq_1 and pq_2 which populate the potential owner roles po_1 and po_2 of the mutual exclusive tasks t_1 and t_2. By cross-assigning the people queries to the excluded owners of the respective other tasks, the two sets of people who can actually execute these tasks are disjoint.

Dynamic task-based SOD: Dynamic task-based separation of duty demands that mutual exclusive tasks must not be executed by the same subject in the same process instance. This concept can be directly represented in B4P/HT by using the getActualOwner method, and assigning it to the excluded owner role of the respective other task.

4.2 Access Control Enforcement Issues in B4P/HT

In the previous section, we discussed how role-based and task-based separation of duty constraints can be captured in B4P/HT. Yet, one needs to be aware of some limitations of this approach, in particular, regarding task delegation and forwarding, the getActualOwner method, and the role of the business administrator:

– The proposal that we make in the previous section can only partially avoid delegation and forwarding of B4P/HT tasks. While the delegation parameters of a task can be explicitly set, it is, according to the current B4P/HT specification, not possible to switch off the forwarding mechanism. Albeit the specification states that "excluded owners are implicitly removed from the set of potential owners", it is unclear when and how this requirement is enforced.

Thus, a potential owner, who is assigned to a task instance in accordance with the SOD constraints, can potentially forward this task instance to an unauthorised user. In general, it is advisable to disallow forwarding in certain scenarios to prevent forwarding operations which would violate the active set of SOD constraints for a task. Therefore, we recommend clarifying the enforcement mechanism or providing means to explicitly switch off forwarding in a future version of B4P/HT.

- Moreover, we encountered an issue with evaluating the getActualOwner method if dynamic task-based SOD constraints are defined for two concurrent tasks. In this case, a task can be started at a point in time when there is not yet an actual owner determined for the other (mutual exclusive) task. Accordingly, the excluded owner lists of both (mutual exclusive) task instances are populated with the empty set (see also Sections 2 and 4.1), and the corresponding dynamic separation of duty constraint is not enforced. Furthermore, at this stage the B4P/HT specification only allows determining the actual owner when the task reaches a final state. From an access control perspective both options – current owner as well as owner history of a task instance – are important and are necessary to make certain access control decisions. Moreover, the current actual owner needs to be known, e.g. in case of escalation. Therefore, it seems to be a good choice to include two different methods in a future version of the specification which can provide these information.
- Finally, the user acting in the business administrator role can override almost all restrictions of a B4P/HT process. While this is critical regarding the level of trust that a subject fulfilling this role would deserve, there is another issue regarding awareness of SOD constraints. A B4P/HT-compliant implementation should provide information about possible SOD violations to the business administrator when he forwards, delegates, or nominates somebody for a task. This way, he should be able to avoid assignments that contradict SOD rules.

Altogether, the result is that B4P/HT offers mechanisms to capture different aspects of SOD constraints. Still, there are some weaknesses regarding enforcement that should be fixed in future versions of the specification.

5 Related Work

While this paper provides the first analysis of the B4P/HT specification from a separation of duty perspective, several approaches exist to extend BPEL or closely related workflow notations with access control means: In [17] we presented an approach to extract RBAC models from BPEL4WS processes. Bertino et al. [5] introduce RBAC-WS-BPEL, a language for authorization policies for business processes defined in BPEL. Furthermore, they introduce the business process constraint language (BPCL) – BPCL is defined as an XML schema and allows for the specification of authorization constraints for BPEL processes, such as separation of duty or binding of duty constraints. Wolter and Schaad [30] extend the business process modeling notation (BPMN) with a means to

model task-based authorization constraints. In particular, they focus on separation of duty constraints to model conflicting roles and/or conflicting tasks. In [27] Thomas et al. propose several BPEL extensions to support user tasks and to define access control requirements of these tasks. They suggest a software component called "people activity manager" (which is, however, unrelated to the B4P/HT specifications [1, 2]) that makes access control decisions for user tasks. We complement this work by analyzing the expressiveness and weaknesses of B4P. In this regard we extend previous work on a workflow resource pattern evaluation of B4P that only briefly touches access control issues [20, 21].

6 Conclusion

In this paper, we analyzed options how the current B4P/HT specifications can support different types of separation of duty constraints. In particular, we discussed how people queries, in conjunction with generic B4P/HT roles, can be used to enforce several aspects of SOD constraints. Moreover, we identified limitations in the current B4P/HT specification and propose solutions to address these limitations in future versions of B4P/HT.

In particular, we suggest providing either a clarification of the enforcement mechanism for excluded owners or a mechanism to explicitly switch off task forwarding in a future version of B4P/HT – this is to prevent users from circumventing SOD constraints via B4P/HT's forward mechanism. Second, the B4P/HT specification only allows to determine the actual owner of a task when the task reaches its final state. However, from an access control perspective it can be important to determine the current owner as well as the owner history of a certain task instance. Thus, we propose to include two different methods in a future version of the B4P/HT specification – one method that returns the current task owner only, and an other method that returns the owner history of a task. Third, because the B4P/HT business administrator role could override almost all restrictions defined for a B4P/HT process, we suggest that B4P/HT-compliant implementations make a user acting as business administrator aware of potential SOD violations when he intends to forward, delegate, or nominate somebody for a task. This way, a user acting in the business administrator role can avoid assignments that contradict SOD rules. An other option could be a restriction of the B4P/HT business administrator role, which, however, may be an undesired option.

In addition to our suggestions summarized above, a native B4P/HT extension would be beneficial to provide explicit (implementation independent) concepts that enable the direct and explicit integration of access control relevant information in B4P/HT task definitions. In our future work, we therefore aim to define such native extensions. Furthermore, when using separation of duty constraints in workflow environments, it is important to enable consistency checks on tasks and corresponding constraint specifications, to guarantee that the constraints properly control task execution and user-to-task assignment without preventing the processes from being completed (see, e.g., [26, 30]) Thus, we also intend

to provide such features for B4P/HT environments in our future work. Finally, there is a need for an efficient and consistent approach to specify access control information in B4P/HT. In future research, we will investigate the suitability of a model-driven approach to generate B4P/HT based on the rules we identify in this paper to express separation of duty.

References

1. A. Agrawal, M. Amend, M. Das, C. Keller, M. Kloppmann, D. König, F. Leymann, R. Müller, G. Pfau, K. Ploesser, R. Rangaswamy, A. Rickayzen, M. Rowley, P. Schmidt, I. Trickovic, A. Yiu, and M. Zeller. Web services human task (WS-HumanTask), version 1.0. 2007.
2. A. Agrawal, M. Amend, M. Das, C. Keller, M. Kloppmann, D. König, F. Leymann, R. Müller, G. Pfau, K. Ploesser, R. Rangaswamy, A. Rickayzen, M. Rowley, P. Schmidt, I. Trickovic, A. Yiu, and M. Zeller. WS-BPEL extension for people (BPEL4People), version 1.0. 2007.
3. G.J. Ahn and R. Sandhu. Role-based Authorization Constraints Specification. *ACM Trans. on Information and System Security (TISSEC)*, 3(4), November 2000.
4. A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland, A. Guizar, N. Kartha, C.K. Liu, R. Khalaf, D. Koenig, M. Marin, V. Mehta, S. Thatte, D. van der Rijn, P. Yendluri, and A. Yiu. Web Services Business Process Execution Language - Version 2.0. OASIS, January 2007.
5. E. Bertino, J. Crampton, and F. Paci. Access Control and Authorization Constraints for WS-BPEL. In *Proc. of the IEEE International Conference on Web Services (ICWS)*, September 2006.
6. E. Bertino, E. Ferrari, and V. Atluri. The Specification and Enforcement of Authorization Constraints in Workflow Management Systems. *ACM Transactions on Information and System Security (TISSEC)*, 2(1), February 1999.
7. R.A. Botha and J.H.P. Eloff. Separation of duties for access control enforcement in workflow environments. *IBM Systems Journal*, 40(3), 2001.
8. D.D. Clark and D.R. Wilson. A Comparison of Commercial and Military Computer Security Policies. In *Proc. of the IEEE Symposium on Security and Privacy*, 1987.
9. D.F. Ferraiolo, J.F. Barkley, and D.R. Kuhn. A Role-Based Access Control Model and Reference Implementation within a Corporate Intranet. *ACM Transactions on Information and System Security (TISSEC)*, 2(1), February 1999.
10. D.F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, and R. Chandramouli. Proposed NIST Standard for Role-Based Access Control. *ACM Transactions on Information and System Security (TISSEC)*, 4(3), August 2001.
11. M.P. Gallaher, A.C. O'Connor, and B. Kropp. The Economic Impact of Role-Based Access Control. National Institute of Standards & Technology (NIST), Planning Report 02-1, March 2002.
12. C.K. Georgiadis, I. Mavridis, G. Pangalos, and R.K. Thomas. Flexible Team-Based Access Control Using Contexts. In *Proc. of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT)*, May 2001.
13. M.A. Harrison, W.L. Ruzzo, and J.D. Ullman. Protection in Operating Systems. *Communications of the ACM*, 19(8), August 1976.
14. H. Koshutanski and F. Massacci. Interactive access control for web services. In Y. Deswarte, F. Cuppens, S. Jajodia, and L. Wang, editors, *IFIP 18th WorldComputer Congress, TC11 19th Int. Information Security Conference*, pages 151–166. 2004.

15. C.E. Landwehr. Formal Models for Computer Security. *ACM Computing Surveys*, 13(3), September 1981.
16. J. Mendling, M. zur Muehlen, and A. Price. *Process Aware Information Systems: Bridging People and Software Through Process Technology*, chapter Standards for Workflow Definition and Execution, pages 281–316. Wiley Publishing, 2005.
17. J. Mendling, M. Strembeck, G. Stermsek, and G. Neumann. An Approach to Extract RBAC Models from BPEL4WS Processes. In *Proc. of the 13th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE)*, June 2004.
18. G. Neumann and M. Strembeck. A Scenario-driven Role Engineering Process for Functional RBAC Roles. In *Proc. of 7th ACM Symposium on Access Control Models and Technologies (SACMAT)*, June 2002.
19. S. Osborn, R. Sandhu, and Q. Munawer. Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies. *ACM Transactions on Information and System Security (TISSEC)*, 3(2), February 2000.
20. N. Russell and W.M.P. van der Aalst. Evaluation of the BPEL4People and WS-HumanTask Extensions to WS-BPEL 2.0 using the Workflow Resource Patterns. BPM Center Report BPM-07-10, BPMcenter.org, 2007.
21. N. Russell, W.M.P. van der Aalst, A.H.M. ter Hofstede, and D. Edmond. Workflow resource patterns: Identification, representation and tool support. In O. Pastor and J. Falcão e Cunha, editors, *17th International Conference CAiSE 2005, Proceedings*, volume 3520 of *Lecture Notes in Computer Science*, pages 216–232. 2005.
22. R.S. Sandhu and P. Samarati. Access control: Principles and practice. *IEEE Communications*, 32(9), September 1994.
23. M. Strembeck. Conflict Checking of Separation of Duty Constraints in RBAC - Implementation Experiences. In *Proc. of the Conference on Software Engineering (SE 2004)*, February 2004.
24. M. Strembeck. A Role Engineering Tool for Role-Based Access Control. In *Proc. of the 3rd Symposium on Requirements Engineering for Information Security (SREIS)*, August 2005.
25. M. Strembeck. Embedding Policy Rules for Software-Based Systems in a Requirements Context. In *Proc. of the IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY)*, June 2005.
26. K. Tan, J. Crampton, and C.A. Gunter. The Consistency of Task-Based Authorization Constraints in Workflow Systems. In *Proc. of the IEEE Workshop on Computer Security Foundations (CSFW)*, June 2004.
27. J. Thomas, F. Paci, E. Bertino, and P. Eugster. User Tasks and Access Control over Web Services. In *Proc. of the IEEE International Conference on Web Services (ICWS)*, July 2007.
28. R.K. Thomas. Team-based Access Control (TMAC): A Primitive for Applying Role-based Access Controls in Collaborative Environments. In *Proc. of the ACM Workshop on Role Based Access Control*, 1997.
29. R.K. Thomas and R.S. Sandhu. Task-based authorization controls (TBAC): A family of models for active and enterprise-oriented authorization management. In *Proc. of the IFIP WG11.3 Conference on Database Security*, August 1997.
30. C. Wolter and A. Schaad. Modeling of Task-Based Authorization Constraints in BPMN. In G. Alonso, P. Dadam, M. Rosemann, editors, *5th Int. Conf. on Business Process Management*, *Lecture Notes in Computer Science*, pages 64–79. 2007.