Integrating Users in Object-aware Process Management Systems: Issues and Challenges

Vera Künzle and Manfred Reichert

Institute of Databases and Information Systems, Ulm University, Germany
{vera.kuenzle,manfred.reichert}@uni-ulm.de

Abstract. Despite the increasing maturity of contemporary Workflow Management Systems (WfMS), there still exist numerous process-aware application systems with more or less hard-coded process logic. This does not only cause high maintenance efforts (e.g. costly code adaptions), but also results in hard-coded rules for controlling the access to business processes, business functions, and business data. In particular, the assignment of users to process activities needs to be compliant with the rights granted for executing business functions and for accessing business data. A major reason for not using WfMS in a broader context is the inflexibility provided by their activity-centered paradigm, which also limits the access control strategies offered by them. This position paper discusses key challenges for a process management technology in which processes, data objects and users are well integrated in order to ensure a sufficient degree of flexibility. We denote such technology as *Object-Aware Process Management System* and consider related research as fundamental for the further maturation of process management technology.

Key words: Object-aware Process Management, Data-driven Processes, Process Design Methods, Access Control, Human Aspects

1 Introduction

Contemporary application systems (e.g., ERP and CRM systems) enable access to *business data*, offer a variety of *business functions* to users, and provide an integrated view on supported *business processes*. However, in most cases underlying business process logic, business functions, and access control constraints are hard-coded within the application system. As a major drawback, even simple process changes then require costly code adaptations and high efforts for testing [1]. To cope with this unsatisfactory situation, *Workflow Management Systems (WfMS)* have been introduced. Usually a WfMS provides generic functions for modeling, executing and monitoring business processes. Contemporary WfMS, however, have not achieved the technological maturity yet for adequately supporting the processes from *Application Systems*. In particular, existing WfMS show strong limitations if a close integration of the process and data perspectives is needed. Another challenge constitutes *access control*. Many information systems rely on *Role-Based Access Control (RBAC)* mechanisms [2, 3] to make

the specification of permissions independent from concrete users and thus to ease users management. However, the basic RBAC approach does not allow to distinguish between permissions of the type and of the instance level [4]. Furthermore, a closer integration of process and data necessitates a more sophisticated approach for access control. On the one hand, access to business data and permissions for executing business functions depend on the executed processes. On the other hand, when defining actor assignments for process activities, permissions for accessing data as well as for executing functions have to be taken into account. Current WfMS do not adequately cope with such interdependencies. In [1] we have introduced an advanced paradigm for the support of *data-driven processes* and have discussed some of the challenges emerging in this context. In particular, we have elaborated the requirements for a generic process management component, which integrates the data- and function-based view, known from many application systems, with a view on the supported processes. Thereby, our specific focus has been on the integration of data and processes. This paper adds a complementary aspect to our previous discussions, namely how to integrate users in *Object-aware Process Management Systems*[1]; i.e., in addition to the challenges arising from a closer integration of process and data, we discuss fundamental requirements for access control in Object-aware Process Management Systems. Understanding these challenges is fundamental in order to integrate users in such systems. We use the following example to illustrate relevant issues for realizing an Object-aware Process Management System.

Illustrating Example. We consider the (simplified) process of a job application as typical in *human resource management*. Using an Internet online form, interested `job applicants` may apply for a `vacancy`. The overall goal of the process is to decide which applicant shall get the offered job. Different `personnel officers` are responsible for different `job applications`. They may request internal `reviews` for each job applicant. Corresponding review forms have to be filled out by `employees from functional divisions` until a certain deadline. Usually, they evaluate the application(s), `make a proposal` on how to proceed (e.g., whether or not a particular applicant shall be invited for an interview), and `submit` their recommendation to the personnel officer. Based on the reviews the personnel officer makes his `decision` on the application(s) or he initiates further steps like interviews or additional reviews. In general, different reviews may be requested and submitted respectively at different points in time.

Section 2 summarizes important aspects on application systems as well as on WfMS, and describes five key challenges for integrating data and processes in Object-aware Process Management Systems. In Section 3 we first present basic issues related to access control. Then we discuss additional challenges which specifically deal with the integration of users. Section 4 discusses existing approaches. The paper concludes with an outlook in Section 5.

[1] An Object-aware Process Management System denotes a process- and object-aware information system with a tight integration of the process and the data perspective.

2 Integrating Processes and Data

In [1] we have introduced the notion of *Object-aware Process Management System*. Our overall goal is to provide a generic component for enabling *data-driven processes* with an *integrated view* on process and data. On the one hand, we want to offer similar features as realized in hard-coded application systems, on the other hand we want to benefit from the advantages known from workflow technology; i.e.; to provide generic functions for realizing such applications. This section summarizes challenges for integrating processes and data in Object-aware Process Management Systems, which we elaborated in two case studies.

2.1 Backgrounds

We first provide the needed backgrounds for understanding following discussions. **Application Systems** provide features for managing business data. The latter are represented by a number of *object instances* (e.g. `job application of Luise Loop`) of different *object types* (e.g. `job application`), which may be related to each other (e.g., for a `job application` several `reviews` may exist). Object types and their relations are modeled in the (application) *data structure*. An object type consists of a set of *attributes* (e.g., `name` of an `applicant` or `decision` about a `job application`). The instances of a particular object type then differ in the values of these attributes. Using basic *functions*, object instances can be created, deleted or changed; i.e, attribute values can be read/written by authorized users. Several object instances can be displayed in a table; whose rows correspond to object instances [1]. Users may invoke specific functions (e.g. forms) on selected object instances in order to read/write its attribute values. In addition to this data- and function-centered view, existing applications often provide an integrated view on processes. Generally, a *process* is needed to reach a particular *business goal*; i.e., object instances having certain attribute values. Therefore, application systems implement *mandatory activities* which have to be performed on these object instances in a certain order and be assigned to authorized users; i.e., *mandatory activities* are executed in the context of a particular process and are added to user worklists. Users are themselves treated as entities maintained in the underlying application database. Consider `applicants`, `personnel officers` and `employees` as users of a human resource management system. Generally, for each real world entity type (e.g. `job application`), and therefore for each role (e.g. `applicant`), a specific object type exists. As a result, each user is represented by an object instance. Consequently, it is not sufficient to assign privileges only on basis of object types and roles. Instead, respective systems have to manage *permissions* at the level of individual object instances as well; e.g. a user may work on a particular process activity for process instance A, while he is not allowed to work on the same activity within another process instance B of same type. Finally, the permissions to work on a particular activity at instance level may depend on accessed data. One drawback of existing applications is the hard-coding of the process logic and the authorization constraints within the application system.

Workflow Management Systems. WfMS offer promising perspectives for realizing more flexible process implementations. Usually a *process definition* consists of a set of activities and their execution constraints (e.g., *control flow*) [5]. The latter can be defined based on a number of control flow patterns which, for example, allow to express sequential, alternative and parallel activity routing or loop backs [6]. Each single *activity*, in turn, represents a task and is linked to a specific business function of an application system. Application data is usually managed by the invoked applications themselves. Only data needed for process control (e.g. for evaluating transition or join conditions) or for supplying activity input parameters of activities are stored and managed within the WfMS. In many WfMS, only *atomic data elements* (i.e. attributes) can be handled; i.e., grouping data elements or defining semantical relations between them is not possible. Roles and users are typically captured in an *organization model* maintained by the WfMS [7]. To be able to assign human activities to the right actors, WfMS use *actor expressions* which are related to components of the organizational model (e.g. user roles). Such assignments have to be defined for each human activity. At runtime, for each business case an instance of the corresponding process definition is created and executed according to the defined control flow. A particular activity may be only enabled if all preceding activities are completed or cannot be executed anymore (except loop backs). When a human activity becomes enabled, corresponding work items are added to worklists of authorized users. Finally, when such a work item is selected by a user, the WfMS launches the associated business function of an application system.

2.2 Basic Challenges for Integrating Processes and Data

Process support in Application Systems raises challenges not adequately addressed by existing WfMS (see [1] for details):

Challenge 1 (Integrating process and data). Processes need to be tightly integrated with application data [8]; i.e.; business data should be managed based on processed objects. Another challenge is to cope with the varying and dynamic number of object instances to be handled within processes during runtime; for each `job application` a different number of `reviews` may exist, which can be instantiated at different points in time. Therefore, the relations between object instances have to be considered during process execution; e.g., to decide about a `job application` only the `reviews` for this concrete application have to be evaluated. In this context, authorized users work on *mandatory activities* needed for the progress of the process instance and offered to users in their worklists. In addition, they may optionally edit attribute values of object instances at arbitrary points in time (denoted as *optional activities*) [1]. As example consider attribute `comment` of a `review` object instance that may be changed at any point in time.

Challenge 2 (Choosing granularities for process and activities). The modelling of processes and data constitute two sides of the same coin and therefore should be compliant with each other [9]. We have to distinguish between

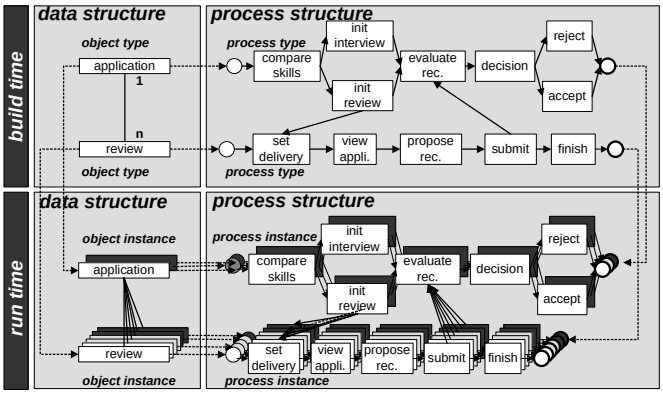

Fig. 1. Analogy between data and process structure

object level and *(data-) structure level*: A particular *process type* should be modeled in accordance with a specific *object type* (see Fig.1). Further, process activities may directly refer to attributes of this object type; e.g., in activity `make decision` of a `job application` the value of attribute `decision` has to filled out. Furthermore, the overall *data structure* has to be taken into account when executing a process instance. To decide on a `job application` the corresponding `reviews` have to be evaluated. Relations between process instances should therefore correspond to the ones between the corresponding data objects [10]. Finally, process instantiation needs to be coupled with the creation of related object instances [1].

Challenge 3 (Data-based modeling). *Process steps* should not be modeled in terms of black-box activities [8], but be defined as *data conditions* on the attributes of the corresponding object type (see Fig. 2); e.g., to reach the process goal of a `job application`, attribute `decision` must be set. This allows us to determine necessary attribute changes over time; i.e., to figure out what attribute changes are required for an object instance to reach the next logical process step of the corresponding process instance. Attribute changes affecting the progress of a process instance can be bound to the execution of *mandatory activities*. Simultaneously, *optional activities* can be executed on the respective object instance (e.g. changing the attribute `comment` of a `review`); but usually have no influence on the progress of the process.

Challenge 4 (Synchronizing process instances). It should be possible to execute both instances of the same and instances of different process types *asynchronously* to each other. However, due to data dependencies at object instance level, we need to be able to synchronize their execution at certain points [1, 10, 11]. Furthermore, to a super-ordinate process instance several sub-ordinate process instances should be assignable in accordance with the relationships between the corresponding object instances and their cardinalities [1]. Fig.1 depicts an example of synchronized process instances. Here, the process instance

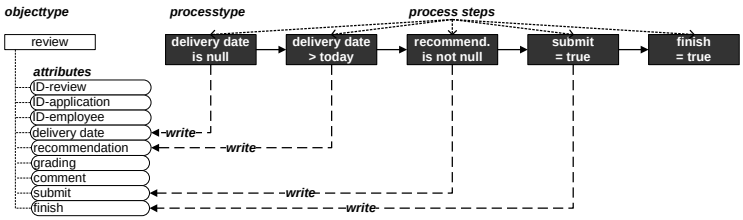

Fig. 2. Data-based modelling

for a `job application` may continue while the process instances for the related `reviews` are executed. Before deciding on a `job application`, however, all requested `reviews` have to be submitted.

Challenge 5 (Flexibility). Process execution is *data-driven*; i.e., it should not be guided by activities, but be based on the state of processed object instances. This way, a more flexible execution behaviour can be realized. In addition to mandatory activities, optional ones may be performed asynchronously. Which optional activities are available at a certain point in time depends on the state of the processed object instance. For example, after submitting a `review` its attribute `comment` cannot be changed anymore. Finally, it should be possible to conjointly work on multiple activities of same type, but belonging to different process instances, i.e. to process the activities in one go [12, 1]. A user should be able to change the values of certain attributes for a set of object instances simultaneously (e.g. to decide on several `job applications`).

3 Integrating Users

This section first summarizes backgrounds on access control. Following this, we discuss fundamental challenges for integrating users in *Object-aware Process Management Systems*.

3.1 Backgrounds on Access Control

Access control mechanisms (so called *authorization*) protect data from unauthorized access (*confidentiality*) and improper changes (*integrity*) [13]. Coincidently, one has to ensure that each user gets access to all required data and functions (*availability*) [13, 14]. We need to consider access control at different layers of abstraction [15]: *strategies, models* and *mechanisms*. *Strategies* determine which components (e.g. data, functions) within a system shall be protected, and define the required kinds of privileges. A *model*, in turn, formally represents the applied strategy, whereas the used *mechanism* determines its technical implementation. Most existing systems use *Role-Based Access Control* (RBAC) as strategy. The additional layer between users and privileges allows for quicker and less cumbersome administration [3]. Furthermore, users with same positions or duties get

same rights [14]. Complementary to these abstraction layers, different kinds of systems make different claims in respect to the needed strategy. Access control can be arranged in four levels, each of them depending on the functionality of the system [16].

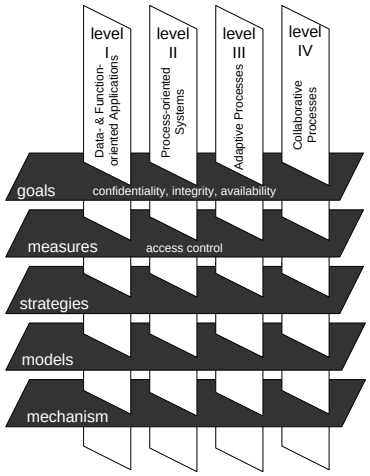

Fig. 3. Classification of access control

Application Systems are on Level 1 and WfMS are assigned to Level 2. Levels 3 and 4, in turn, are used for process-aware information systems enabling adaptive or collaborative processes [17]. Fig. 3 shows the different abstraction layers and the different levels of access control. In the following we focus on systems covering Levels 1 and 2. We only consider the abstraction layer of the strategy dimension. Technical aspects (e.g., models or concrete mechanisms) are out of the scope of this paper. Further, we limit ourselves to privileges for executing activities and for reading application data.

3.2 Challenges for Integrating Users

Challenge 6 (Horizontal and vertical user assignment). In WfMS, human activities are associated with actor expressions; e.g., based on user roles (denoted as *horizontal authorization*). Users who may work on respective activities are then determined during runtime based on these expressions. This is no longer sufficient in our case, since the selection of potential actors does not only depend on the activity itself, but also on the object instance processed by this activity [18, 19] (denoted as *vertical authorization*). Consider Fig. 4, which depicts a partial view on different process instances handling `job applications`. Fig. 4a illustrates horizontal authorizations, whereas Fig. 4b shows horizontal as well as vertical authorization. While certain actors are only allowed to perform activity `make decision` for applicants whose name starts with a letter between 'A' and 'L', other ones may only perform this activity for applicants whose name starts with a letter between 'M' und 'Z'. Existing WfMS fail to deal with such data-dependent, vertical authorizations.

Challenge 7 (Consistency between data- and process authorization). For *optional activities* it does not make sense to obey a strict execution order as for *mandatory* ones. Although the progress of a process instance is based on attribute values, changes of these values should be possible outside the scope of normal process execution as well. However, when executing such *optional activities*, undesired manipulations of object attributes have to be prevented. After an employee from a functional division has submitted his `review`, for example he is no longer allowed to change the value of attribute `recommendation`. For

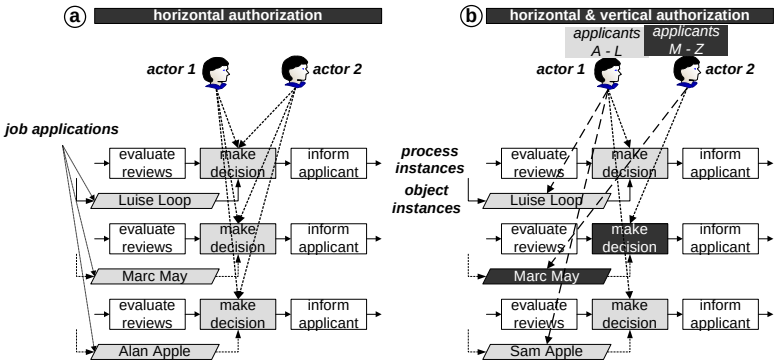

Fig. 4. Horizontal and vertical user assignment

this reason, optional activities cannot be handled completely independent from mandatory ones; i.e., authorization for optional activities of an object instance needs to consider the progress of the corresponding process instance [20]. In particular, correct executability of the process instance has to be guaranteed; i.e., it must be ensured that all mandatory activities of the process can be executed. Since a process step is defined based on attribute values, the user who is executing the mandatory activities should therefore have the permission to change corresponding attributes. Fig. 5 shows an object instance and process instance respectivley as well as the corresponding privileges for a `review`. Permissions to read / write a specific attribute value depend on the progress of the corresponding process instance. Attributes whose values have to be changed within a mandatory activity are marked with black; attributes that can be read or changed when executing optional activities are coloured white. Note that the latter may vary depending on the progress of the process instance. Users are not allowed to change attribute values used for the definition of a previous process step (except loops). These permissions are coloured grey.

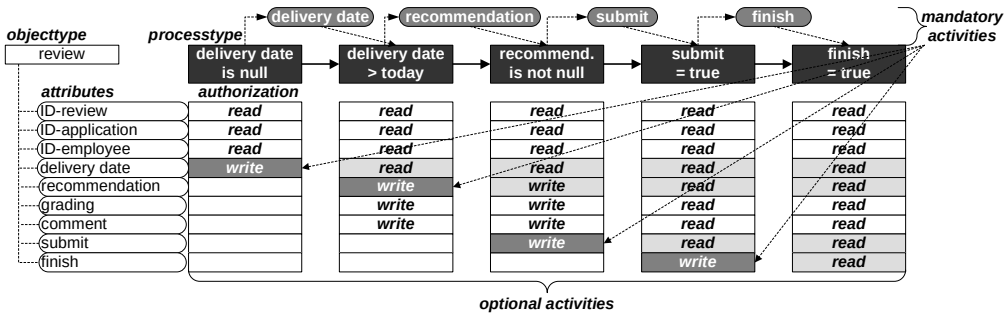

Fig. 5. Consistency between data- and process authorization

Challenge 8 (Relations between users and object instances). As motivated, *horizontal* as well as *vertical* authorizations are needed. Regarding *vertical authorizations*, permissions have to be restricted to a set of object instances or one particular object instance [21, 22]. Thereby, the mapping between users and object instances (i.e., the regulation which user has which access rights for which object instances) is not arbitrary, but underlies certain conditions [23]. An `applicant` is allowed to read his own `job application`, but not the `job applications` of other `applicants`. Therefore, it is not appropriate to treat actor assignments and application data independent from each other as in existing WfMS. To achieve this, data structures (i.e., object types and their relations to each other) should also include organizational entities; i.e., roles have to be explicitly defined as object types. Each individual user is then mapped to an object instance. At runtime, a user may reference other object instances; e.g., an organisation unit. Users themselves may be referenced by other object instances; e.g., an applicant may submit several applications which then refer to this user. The different relationships between users and object instances have to be taken into account when assigning actors to activities.

Challenge 9 (User assignment and authorization). Permissions are defined based on the attributes of object types. When executing mandatory activities, attribute changes necessary for enabling the next process step have to be accomplished. In order to assign users to mandatory activities, their permissions are evaluated. Having the permission to change certain attributes does not necessarily mean that the user has to perform a mandatory activity; i.e., we have to distinguish between mandatory and optional permissions. Only for users with mandatory permissions, a respective, mandatory activity is listed in their worklist. Users with optional permissions may change the corresponding attributes when executing optional activities. This means, users may change the attribute values, relevant to enable the next process step, when executing an optional activity. Such implicit transitions may be desired in some situations, while in other cases responsible users should explicitly verify the specified attribute values; e.g., attribute `proposed action` can be entered when executing an optional activity. As effect, the next process step `filled out` can be reached without having executed mandatory activity `fill out`. By contrast, in order to `submit` a `review`, mandatory activity `submit` will have to be executed even if attribute `submitted` has been entered within an optional activity (see Fig.5).

4 Existing Approaches

In [1] we have described existing approaches in relation to Challenges 1 - 5. In particular, some of the discussed issues are addressed in Artifact-Centric Modelling [24], Product-Based Workflow-Support [9, 25], Data-Driven Process Coordination [10, 26], Case Handling [8], and Proclets [11]. In the following, we discuss existing work in respect to Challenges 6 - 9.

Challenge 6 (Horizontal and vertical user assignment). [27] describes an approach for realizing Applications Systems, which groups permissions for

accessing data and functions. Whether a user may perform a particular activity depends on the agreement of another user at runtime. This makes it possible to manually approve the object instances relevant in the given situation. However, it is not possible to access data outside the scope of a specific activity; i.e., optional activites are not considered. Like [27], current WfMS focus on actor assignments for controlling activity executions. By contrast, permissions for accessing data and functions are mostly managed within the invoked application systems. [28] describes the concept of "instance-based user group". Each actor gets access to all object instances associated with at least one process instance he has been involved in. The opposite direction (i.e., user assignments depending on permissions for data access) is not considered. [18] enables management of specific properties for each data element relevant for the process. In addition to actor expressions, for each activity, relevant properties of the used data elements are defined. Obviously, this is done redundantly and therefore data inconsistencies might arise.

Challenge 7 (Consistency between data and process authorization). [8] distinguishes between mandatory and optional data elements of an activity; but no differentiation between mandatory and optional activities is made. Similar to [28], users get access to all data elements of the process instances they are involved in (i.e., the permission to read / write data is assigned implicitly). [27] and [20] define permissions for accessing data and functions in the context of a specific task. The other direction (i.e., the assignment of users to tasks depending on given permissions on functions) is not considered. Since all permissions are defined at the level of object types, it is not possible to assign different permissions for object instances of the same type.

Challenge 8 (Relations between users and object instances). In some approaches [29, 22, 30, 21, 23], it is possibe to restrict permissions to a selected set of object instances. However, only in few cases [22, 21, 23] these restrictions can be defined depending on the relationships between users and object instances. In particular, it is not possible to consider relationships already defined in the data structure. Instead, they have to be defined redundantly based on the permissions.

Challenge 9 (User assignment and authorization). Except few systems (e.g., case handling) WfMS do not support optional activities as described in sect. 3. Hence, it is not possible to differentiate between tasks users must execute and tasks they may execute. In [31], various possibilities for assigning and activating activities are described. [3] takes the hierarchy of roles into account and [32] allows to define different priorities for assigning users to activities. However, in all approaches, always at least one user has to execute an activity. Opposed to this, [8] focuses on data access rather than on assigning users to activities. Further, for each activity it is possible to differentiate between optional and mandatory data elements.

In summary, the described challenges have been partially addressed by existing work. However, a comprehensive solution for generic access control in object-aware process management is still missing.

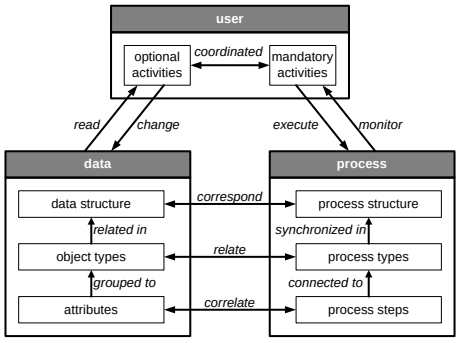

Fig. 6. integration of processes, data and users

5 Outlook

Our overall vision is to develop a framework and enabling technology for *Object-aware Process Management*. We will tackle the described challenges in order to enable a tight integration of processes, data objects and users. Fig.6 illustrates the discussed interdependencies. This work has been conducted within the PHILharmonic Flows project. [2] In future papers we will provide detailed insights into the different components of an Object-aware Process Management system as well as their complex interdependencies.

References

1. Künzle, V., Reichert, M.: Towards Object-aware Process Management Systems: Issues, Challenges, Benefits. In: Proc. BPMDS'09. LNBIP 29 (2009) 197–210
2. Osborn, S., Sandhu, R., Munawer, Q.: Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies. ACM Trans. Inf. Syst. Secur. **3**(2) (2000) 85–106
3. Bertino, E., Ferrari, E., Atluri, V.: The Specification and Enforcement of Authorization Constraints in Workflow Management Systems. ACM Trans. Inf. Syst. Secur. **2**(1) (1999) 65–104
4. Sandhu, R., Coyne, E., Feinstein, H., Youman, C.: Role-based Access Control Models. IEEE Computer **29**(2) (1996) 38–47
5. Aalst, W., Hee, K.: Workflow-Management - Models, Methods and Systems. MIT Press, Cambridge (2004)
6. Aalst, W., Hofstede, A., Kiepuszewski, B., Barros, A.: Workflow Patterns. Distr. & Parallel Databases **14** (2003) 5–51
7. Rinderle-Ma, S., Manfred, R.: A Formal Framework for Adaptive Access Control Models. Journal on Data Semantics IX (2007) 82–112
8. Aalst, W., Weske, M., Grünbauer, D.: Case Handling: A new paradigm for business process support. Data and Knowledge Engineering **53**(2) (2005) 129–162

[2] Process, Humans and Information Linkage for harmonic Business Flows

9. Reijers, H., Liman, S., Aalst, W.: Product-based Workflow Design. Management Information Systems **20**(1) (2003) 229–262
10. Müller, D., Reichert, M., Herbst, J.: Data-driven Modeling and Coordination of Large Process Structures. In: Proc. CoopIS'07. LNCS 4803 (2007)
11. Aalst, W., Barthelmess, P., Ellis, C., Wainer, J.: Workflow Modeling using Proclets. In: Proc. CoopIS'00. (2000) 198–209
12. Sadiq, S., Orlowska, M., Sadiq, W., Schulz, K.: When workflows will not deliver: The case of contradicting work practice. In: Proc. BIS'05. (2005)
13. Bertino, E.: Data security. Data Knowl. Eng. **25**(1-2) (1998) 199–216
14. Ferraiolo, D., Kuhn, R.: Role-based Access Control. In: Proc. 15th NIST-NCSC. (1992) 554–563
15. Samarati, P., Vimercati, S.: Access Control: Policies, Models and Mechanisms. FOSAD (2001)
16. Pfeiffer, V.: A Framework for Evaluating Access Control Concepts in Workflow Management Systems. Master thesis (2005)
17. Weber, B., Reichert, M., Wild, W., Rinderle, S.: Balancing Flexibility and Security in Adaptive Process Management Systems. In: Proc. CoopIS'05 LNCS 3760. (2005)
18. Rosemann, M., Mühlen, M.: Modellierung der Aufbauorganisation in Workflow-Management-Systemen: Kritische Bestandsaufnahme und Gestaltungsvorschläge. EMISA Forum **3**(1) (1998) 78–86
19. Rosemann, M., Mühlen, M.: Organizational Management in Workflow Applications: Issues and Perspectives. Inf. Technol. and Mgmt. **5**(3-4) (2004) 271–291
20. Botha, R.: Cosawoe A Model for Context-sensitive Access Control in Workflow Environments. PhD thesis (2002)
21. Hu, J., Weaver, A.: A Dynamic, Context-Aware Security Infrastructure for Distributed Healthcare Applications. In: Proc. PSPT2004. (2004)
22. Kumar, A., Karnik, N., Chafle, G.: Context Sensitivity in Role-based Access Control. SIGOPS **36**(3) (2002) 53–66
23. Barkley, J., Beznosov, K., Uppal, J.: Supporting Relationships in Access Control Using Role Based Access Control. In: Proc. RBAC'99. (1999) 55–65
24. Liu, R., Bhattacharya, K., Wu, F.: Modeling Business Contexture and Behavior Using Business Artifacts. In: Proc. CAiSE'07. (2007) 324–39
25. Vanderfeesten, I., Reijers, H., Aalst, W.: Product-based Workflow Support: Dynamic Workflow Execution. In: Proc. CAiSE'08. LNCS 5074 (2008) 571–574
26. Müller, D., Reichert, M., Herbst, J.: A New Paradigm for the Enactment and Dynamic Adaptation of Data-driven Process Structures. In: Proc. CAiSE'08. LNCS 5074 (2008) 48–63
27. Sandhu, R., Thomas, R.: Task-based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-oriented Authorization Management. In: Proc. IFIP'97. (1997) 166–181
28. Wu, S., Sheth, A., Miller, J., Luo, Z.: Authorization and Access Control Of Application Data In Workflow-Systems. JIIS **18** (2002) 71–94
29. Lupu, E., Sloman, M.: A Policy Based Role Object Model. In: Proc. EDOC'97. (1997) 36–47
30. Thomas, R.: Team-based Access Control (TMAC): A Primitive for Applying Role-based Access Controls in Collaborative Environments. In: Proc. RBAC'97. (1997) 13–19
31. Russell, N., Hofstede, A., Edmond, D.: Workflow Resource Patterns. In: Proc. CAISE'05. (2005)
32. Wainer, J., Barthelmess, P., Kumar, A.: W-RBAC - A Workflow Security Model Incorporating Controlled Overriding of Constraints. IJCIS **12** (2003) 2003