The Proviado Access Control Model for Business Process Monitoring Components

Manfred Reichert[1], Sarita Bassil[2], Ralph Bobrik[3], Thomas Bauer[4]

[1]Institute of Databases and Information Systems, Ulm University, Germany
manfred.reichert@uni–ulm.de
[2]Computer Science Department, Marshall University, USA
bassil@marshall.edu
[3]Detecon AG, Switzerland
ralph.bobrik@detecon.com
[4]Group Research and Advanced Engineering, Daimler AG, Germany
thomas.tb.bauer@daimler.com

12th August 2010

Abstract

Integrated process support is highly desirable in environments where data related to a particular business process are scattered over distributed, heterogeneous information systems. A business process monitoring component is a much-needed module in order to provide an integrated view on all these process data. Regarding process visualization and process data integration, access control (AC) issues are very important but also quite complex to be addressed. A major problem arises from the fact that the involved information systems are usually based on heterogeneous AC components. For several reasons, the only feasible way to tackle the problem of AC at the process monitoring level is to define access rights for the process monitoring component, hence getting rid of the burden to map access rights from the information system level. This paper presents the Proviado process visualization framework and discusses requirements for AC in process monitoring, which we derived from our case studies in the automotive domain. It then presents alternative approaches for AC: the view-based and the object-based approach. The latter is retained, and a core AC model is proposed for the definition of access rights that meet the derived requirements. AC mechanisms provided within the core model are key ingredients for the definition of model extensions.

1 Introduction

In order to streamline their way of doing business, today's companies are dealing with a large number of processes involving different domains, organizations, and groups (Weske, 2007; Mutschler et al., 2008). As discussed by Bobrik et al. (2005), an integrated process support is highly desirable in such an environment where data (e.g., business data, audit trails and reports) related to a particular process (instance), and with different degrees of sensitivity, are often scattered over heterogeneous information systems (IS) (cf. Fig. 1). A process monitoring component is a much-needed module in order to provide an integrated and abstracted view on all these data (Junginger et al., 2004; Muehlen, 2001; Polyvyanyy et al., 2009). Despite its importance, many existing process-aware information systems do not offer such component. For example, a process monitoring component is specifically responsible for displaying the status of process instances (McGregor and Kumaran, 2002), for dispatching specific activities to corresponding actors (Rinderle and Reichert, 2005), for providing an integrated view on process and application data (Rinderle et al., 2006), or for enabling business performance monitoring (Costello and Molloy, 2008; McGregor, 2002; Muehlen, 2001; Junginger et al., 2004).

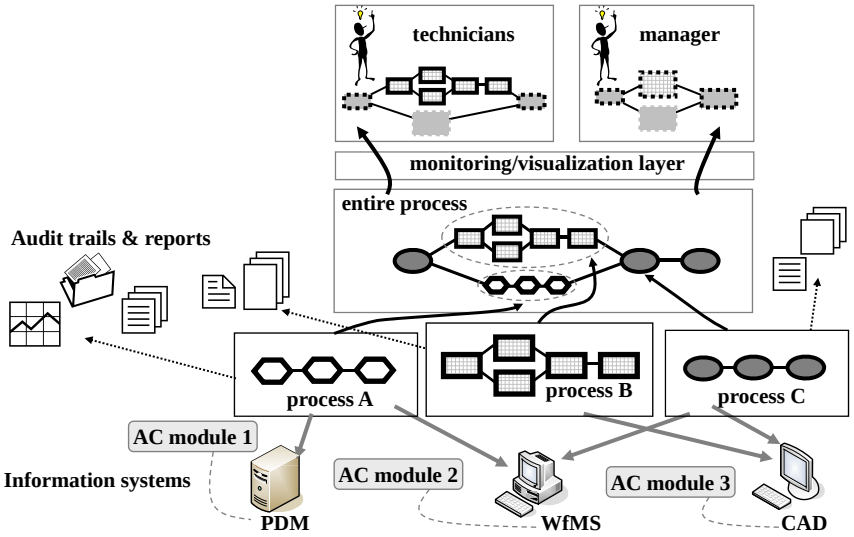

Figure 1: Process Data Integration with Multiple Perspectives

1.1 Problem Statement

Different user groups or roles (e.g., technicians, engineers, managers) usually have different perspectives over processes and related data. In this context,

Bobrik et al. (2007), Polyvyanyy et al. (2008) and Reijers et al. (2009) suggest providing adequate views for the different user groups. This is of particular importance when dealing with complex, long-running business processes with dozens up to hundreds activities (see Fig. 2 for an example from one of the projects we conducted in the automotive domain). Regarding process data integration and process monitoring (Junginger et al., 2004; Polyvyanyy et al., 2009), in addition, access control (AC) issues are very important to be addressed, but have been neglected in existing approaches so far. In this context, a major problem is that involved IS are usually based on different AC components implying facts such as

- heterogeneity regarding the meta-models based on which organizational models and related access rights are defined (e.g., users / groups and actors / roles),
- different notions for the same entity/entity type (e.g., user and actor), and
- non-registration of particular user(s) in all of the involved IS.

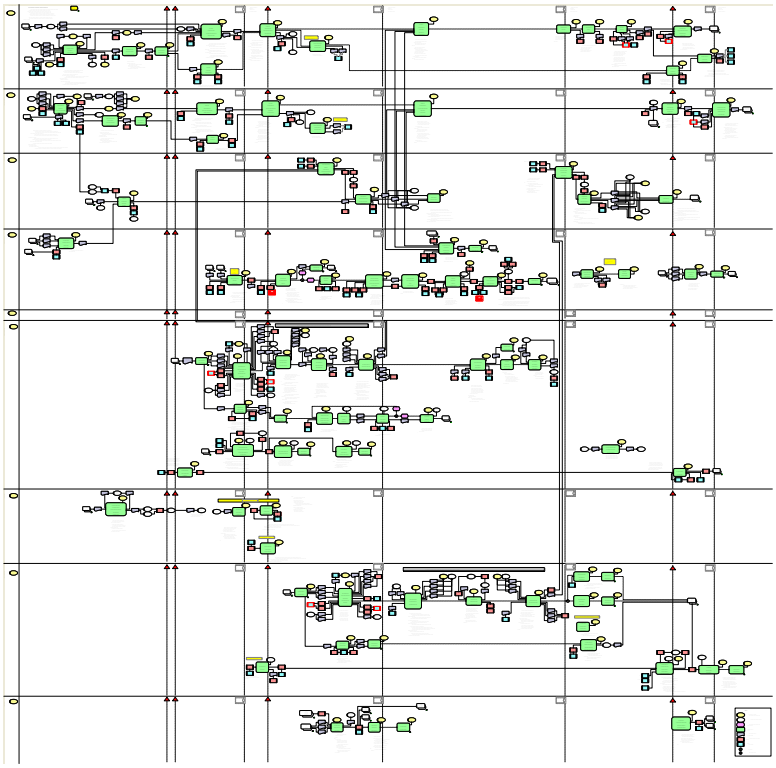

Figure 2: Model of a Complex Engineering Process (Partial View)

To preserve integrity of AC information, AC constraints applied at the process monitoring level should be consistent with the constraints set out by the different IS. However, in our case studies it has turned out that the integration of heterogeneous AC components is difficult to achieve for several reasons:

1. Access rights are not always explicitly described, but might be "hard-coded", and hence difficult to retrieve;

2. AC modules do not always provide (application programming) interfaces in order to facilitate the access to information about AC rules ("black-box" AC modules); and

3. Rights at the IS level mainly deal with process definition and execution, and have been not designed for the monitoring of process data by different users. Process definition and execution require administration rights, permissions to create new instances, rights to work on specific activities (Wainer et al., 2003), delegation rights (Wainer et al., 2007), and rights to change processes (Weber et al., 2005). By contrast, monitoring requires rights to visualize specific process activities, to display specific activity attributes, to visualize application data in the context of active process instances, or to show different abstractions on a process (cf. Fig. 3a+b).

Taking this into account, the only feasible way to tackle the problem of AC at the process monitoring level is to (re-)define AC rights for the process monitoring component, hence getting rid of the burden to inherit AC rights from the IS level. Of course, if possible, existing AC rights at the IS level should be automatically mapped to the ones at the process monitoring level, but we cannot assume this in general. Explicitly, specifying AC rights at the monitoring level also makes it possible to define them at a finer-grained level when compared with what is already defined at the IS level.

1.2 Contribution

The AC approach presented in this paper was developed in the Proviado project (Bobrik et al., 2005, 2006, 2007). Proviado proposes a solution for visualizing in a secure way data related to a particular process or to a collection of processes. This paper significantly extends the work we presented in (Bassil et al., 2009). We give additional insights into our process visualization framework, describe an AC module for it, provide an evaluation of this AC module, and elaborate related work in more detail.

We first discuss issues relevant for the realization of a process visualization (monitoring) component in general as well as requirements for the definition of related AC rights in particular. These requirements have resulted from case studies we conducted in the automotive domain. Amongst others we

analyzed processes in areas like automotive engineering, release management, change management, vehicle repair, and production planning. We discuss two alternative approaches for AC, mainly a *view-based* and an *object-based* one. The retained solution (i.e., the object-based approach) is used as backbone in order to provide a comprehensive core AC model. This model allows for the (compact) definition of AC rights at a fine-grained level. Moreover, AC rights are meant to meet the spectrum of confidentiality possibly defined on process data. Proposed AC mechanisms will be key ingredients in future definitions of extended AC models for process monitoring.

The remainder of this paper is organized as follows: Section 2 sets the context of our research and introduces the Proviado visualization framework, but without including AC issues. Section 3 then exposes the major AC requirements to be met by such a component. Two alternative approaches for AC are studied and compared in Section 4. In Section 5, we introduce our logical AC model. Section 6 provides an evaluation of our approach and Section 7 discusses related work. Finally, Section 8 concludes with a summary and an outlook.

2 The Proviado Approach

This section sets the context of our research. It first introduces basic notions by distinguishing between model and instance level. Then we exemplarily show how sophisticated process visualization is realized in Proviado. For illustrating purposes, we consider a real case from one of our projects in the automotive domain.

2.1 Basic Considerations

Generally, we distinguish between model and instance level (cf. Fig. 3). The former gathers different kinds of enterprise models such as organizational models, functional models, data models, IT-system models, and process models. Each of the first four models gives input to the process model defined as a set of one or more linked activities, which collectively realize a particular business objective. Specifically, these activities are carried out in a coordinated way by different processing entities (including humans and software agents) to reach a goal, such as changing the design of a car, delivering merchandise, or operating a patient. User- and pre-defined attributes may be associated with process models or activities (e.g., costs, needed resources). Examples of frameworks supporting the integrated modeling of the different enterprise aspects include ArchiMate (Groenewegen et al., 2010), ADONIS (Kühn et al., 2003) and ARIS (Davis, 2008).

In Proviado (Bobrik et al., 2005, 2006, 2007), at the *model level*, we focus on the secure visualization of data related to a particular process model. As example consider the model of a *change request process* as it can be

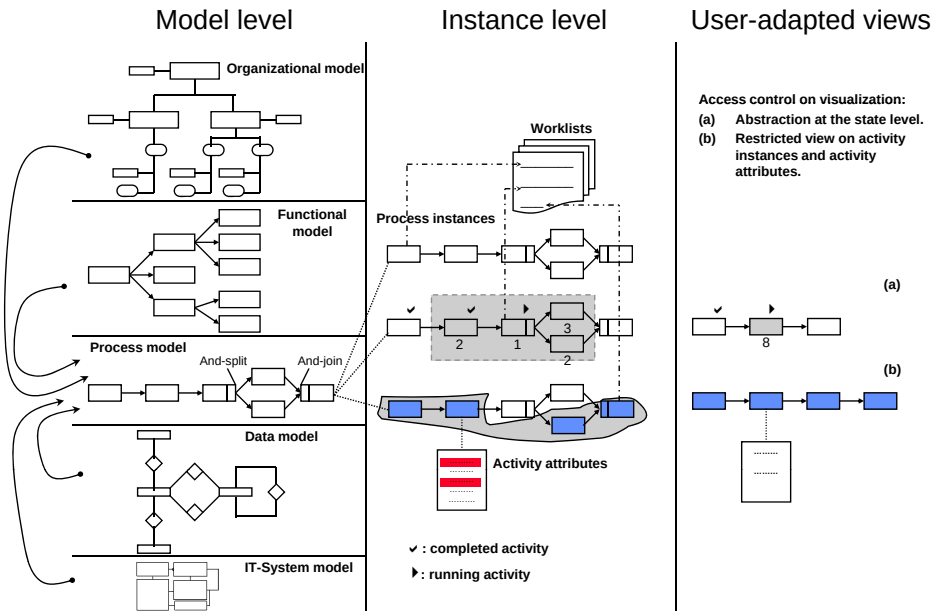

Figure 3: Basic Considerations

found in the automotive domain (cf. Fig. 4). This process model comprises five phases with 20 activities in total. Furthermore, control and data flow, exceptional paths, role assignments, and IT system resources are depicted. Using this example, we will show how a process model can be enriched with instance data and then be displayed to authorized users. Thereby, Proviado enables flexible configuration and personalization of the generated process visualizations.

Other kinds of models have not been considered for visualization yet, but will be added later on. Different types of data may be involved in a process model such as process relevant data and application data (Weske, 2007). We are particularly interested in providing a secure way to visualize application data. These data are in general strictly managed by the application(s) supporting the process model.

At the *instance level*, we focus on the secure monitoring of running process instances. A process instance is defined as the representation of a single enactment of a process model (i.e., a concrete business case). Concepts such as user worklists (i.e., lists of work items derived from process instance activities), activity execution state (e.g., `Running`), and activity execution cost are associated with the instance level.

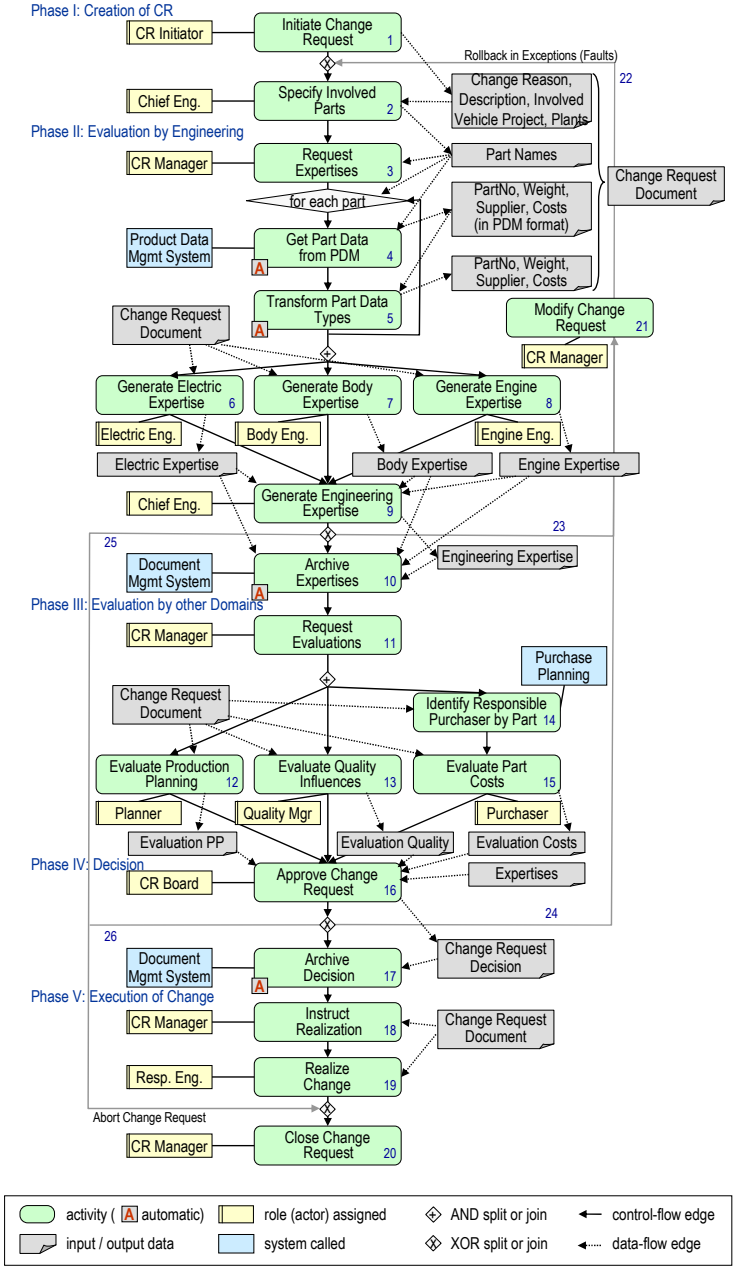

Figure 4: Process Model of a Change Request (CR)

2.2 Process Visualization with Proviado

The Proviado framework targets at flexible and configurable visualizations of business processes. In particular, these visualizations should to be adaptable to the needs of different user groups (i.e., business performers) along the following three dimensions: First, it must be possible to reduce process complexity for users by discarding or aggregating information which are not relevant in the given context or for which the user does not have sufficient access rights. Second, the appearance of process elements (e.g., activities, data objects, control and data connectors) must be customizable independent from the representation of the source process model. Third, different diagram types (e.g., process graph, swim lane, calendar, Gantt diagram, table) should be supported.

2.2.1 Process View Concept and Template Mechanism

For realizing a particular drawing of a process model and process instance respectively, a visualization model can be specified separately from the process. Among other things, such visualization model comprises parameters for configuring which process elements are to be displayed and which notation shall be used. This configuration can be specified at a high level of abstraction based on a powerful view concept and a flexible template mechanism.

Proviado View Concept. The *process view* concept we developed in Proviado allows reducing the complexity of a business process visualization. This is achieved by applying well-defined transformation rules based on process graph reduction and process graph aggregation respectively. The *reduction* operation can be used to remove process objects from a process model. As example consider Fig. 5 where activities `E`, `F` and `G` are removed from the given process model and a new control edge is inserted instead. Fig. 5 also gives an idea of the *aggregation* operation. `Aggregate(B,C,H,K)`, for example, aggregates four activities by replacing them with one abstract node in the process graph. Depending on the concrete structure of the sub-graph induced by the set of activities to be aggregated, different graph transformations may have to be applied. While in some cases the aggregated process view can be realized by simple graph transformations, in other scenarios this necessitates a more complex restructuring of the process graph. Generally, aggregated process views are more difficult to realize than reduced ones. In particular, relations to satellite objects (e.g., data elements, org. roles) have to be preserved (cf. Fig. 5) and attribute values for the abstract activity node resulting from the aggregation have to be calculated. Finally, aggregation operations are provided for all process aspects including data flow, and actor assignments.

It is important to mention that view building operations as provided by Proviado maintain a sound process model if desired. However, to introduce

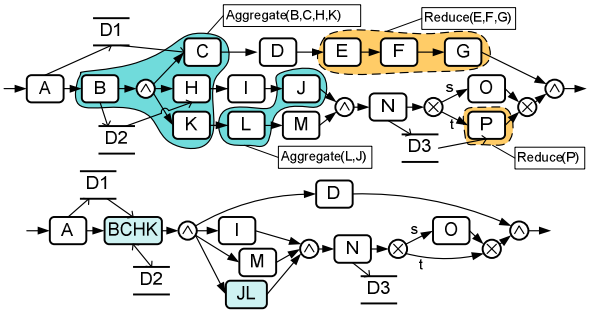

Figure 5: Proviado View Concept

additional flexibility for process visualizations, operations "violating" structural model constraints (e.g. `DeleteEdge`) are considered as well. Higher level operations built on top of aggregation and reduction operations exist that automatically derive the set of activities to be processed. This facilitates maintenance of view definitions when changing the process models they are based on.

Proviado Template Mechanism. While the described view concept allows us to define which process elements shall be displayed, the *Proviado Template Mechanism* (for details see (Bobrik et al., 2006)) enables us to configure the graphical appearance of the different process elements. In this context a *template* represents the concrete notation (i.e., the symbols) to be used for visualizing a particular process element (e.g., an activity or a data object). Its graphical appearance (e.g., shape, arrow) is described based on Scalable Vector Graphics (SVG). By using this XML-based format, to a large degree, we can define templates graphically with a standard SVG Editor.

Each template comprises a set of data fields (i.e., parameters) which can be filled with concrete process data values (e.g., activity name or state) at visualization time. We use XPath expressions to establish the relationship between symbol definition and data fields. Required data transformations (e.g. date format conversion) can be realized via ECMA-Script expressions. Altogether, a complete notation for process visualization consists of a set of templates. More precisely, each process element has to be linked to a template. This link can be established statically (i.e., remain unchanged) or dynamically based on selected process data (e.g., the runtime status of the process element). The latter enables, for instance, to use different symbols for activities, e.g., depending on their state or on the actor working on them. Finally, *Cascading Style Sheets* are used to vary the look of process drawings.

All in all the sketched *Template Mechanism* enables us to use a process notation in an unambiguous and easy to maintain manner. In combination with the view concept personalized process visualizations become possible. While non-relevant process elements can be removed or aggregated with

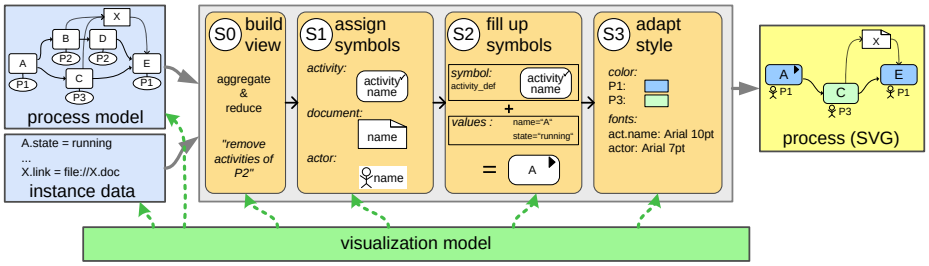

Figure 6: Generating a Process Visualization in Proviado

other objects, the visualization of relevant process elements can be adapted to specific user or application needs.

2.2.2 Configuring a Process Visualization

Fig. 6 shows the basic steps necessary to automatically generate a process visualization. Starting point is an integrated process model, which correlates (fragmented) process data from different source systems in an harmonized way. First, we restrict this visualization content to that information needed by the user (S0). This is realized by a view component which applies aggregation and reduction techniques to process models. Step S0 is followed by formatting steps S1, S2 and S3: S1 fixes the graphical symbols designed for the different process elements. Thereby we consider information from a *visualization model*; S2 fills graphical symbols with real attribute values related to the process model or process instance to be displayed; within S3 formatting parameters are customized to user preferences, e.g., by coloring the process visualization in accordance to cooperate identity guidelines.

2.2.3 Application Example

Consider again the process model from Fig. 4. Assume that an instance of this process shall be visualized for an actor from the engineering domain. For this purpose non-relevant process elements have to be discarded. Automated steps for transforming and exchanging data (e.g. Steps 4 and 5), for example, shall be not displayed. The same applies to selected interactive steps (e.g. Steps 2 and 3). Finally, control edges capturing forward and backward jumps shall be removed. Altogether this process view can be realized by applying the following view operations (listed in brackets for each operation):

Aggregation:[1] {1, 2}, {11, 12, 13, 14, 15}
Reduction: {3}, {4, 5}, {10}, {17, 18}, {20}, {21}

[1]Each operation is listed in brackets. The aggregations result in the activities "Request Creation" and "CR Evaluation"

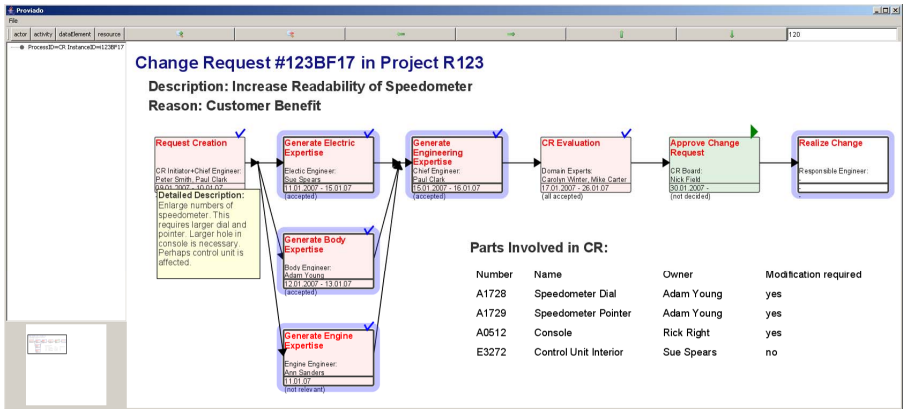

Figure 7: Visualization of a CR Process Instance for Engineers

DeleteEdge: {22, 23, 24}, {25, 26}

The resulting process view would still contain a large number of satellite nodes (representing actors, systems, etc.) which usually shall not be displayed. Our visualization model allows to omit such nodes and to assign their data values to other visualization objects, e.g., activity boxes (cf. Fig. 7). Furthermore, with the *Proviado Template Mechanism* any desired appearance of the process view to be displayed can be realized. For example, the visualization from Fig. 7 contains information like change reason, change description, and involved parts. Furthermore, a header has been added. Other data like a detailed CR description can be accessed via a tool tip. Finally, activities which are of particular importance for engineers are highlighted.

Note that the created process drawing as depicted in Fig. 7 constitutes one possible abstracted visualization of the process model from Fig. 4. Depending on specific user requirements, for example, Proviado allows to provide different visualizations of the same process view, e.g., using a standardized notation like BPMN. Basic to this exchangability of visual representations is the described *Proviado Template Mechanism*. Generally, different information and layouts can be presented. Furthermore, new process views (with same or different appearance) can be easily realized. For example, for managers each of the five phases of the CR process could be aggregated to one single activity and only information about deadlines, delays, resources, and the final decision be visualized (cf. Fig. 8)

We now have to additionally consider access rights in respect to visualized process data. Generally, at model and instance levels, different kinds of rights need to be defined; e.g., administration rights, data access rights, permission to create instances from a given process model, rights to execute a particular work item, or delegation rights. At the model (instance) level, the visualization (monitoring) of user-adapted views (see above) derived

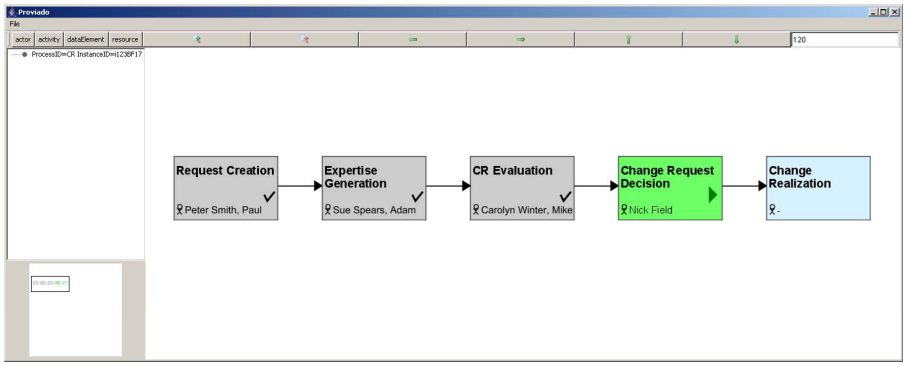

Figure 8: Visualization of a CR Process Instance for Managers

from specific process models (instances) is required. These views must take into account access rights of the involved user. Access rights may be defined on different aspects related to the model and instance levels; e.g., process model, activity, process instance, activity instance, data elements, pre- and user-defined attributes, attribute current value, and attribute history. In the following we discuss major requirements for access control in process monitoring and then present the access control model used in Proviado.

3 Access Control Major Requirements

We conducted case studies in the automotive domain in which we studied processes relating to *car engineering*, *change management*, *vehicle repair*, and *release management* (Müller et al., 2006; Bobrik, 2008). We complemented this by also considering a large number of processes from the healthcare domain (Lenz and Reichert, 2007). As fruit of these case studies, we derived major requirements for AC in process monitoring.

Requirement 1 (Definition of AC rights at a fine-grained level). AC rights for process monitoring should meet the spectrum of confidentiality defined on data related to a particular process. Moreover, they should be definable on different aspects/objects of the model and instance levels (e.g., the process itself and its activities, attributes, and data elements).

Requirement 1.1 (Meeting a spectrum of confidentiality). A distinction should be made between at least three levels of confidentiality: a first level in which all available information can be accessed, a second one where only high-level information can be accessed, and a third one where no information is available at all. We provide some examples to illustrate this:

- **Example 1**. Considering the (simplified) process of managing change requests (cf. Fig. 9a), for example, we may think about a (pre-defined)

attribute (e.g., *activity cost*) associated with a specific activity (e.g., `generate expertise`). Such an activity may require a "two days by person" cost[2] to be accomplished. One may have the right to access this information (i.e., the exact value of the attribute), to access abstracted information such as "less than one week (i.e., less than five days by person)", or to access nothing.

- **Example 2**. Another example could be the "costs" for applying a change to a car. "Costs" may be modeled as an output data element of the *Generate Expertise* activity. Again, the three levels of confidentiality discussed above may be applied in order to access either the exact value assigned to "costs" (e.g., 12.875 Euros), or an approximate value (e.g., less than 15.000 Euros), or to completely hide the information.

- **Example 3**. The spectrum of confidentiality may also be restricted to only two levels: "give" or "don't give information". In change management, for example, an external partner may design part of the car; internally, a verification of this component may be done before it is integrated with the overall design of the car. The external partner might or might not have the right to know about the *existence* of the verification activities.

- **Example 4**. Specific data (e.g., business/technical documents) may be given as input to activities such as `generate expertise`, `provide evaluation`, or `provide comments`; one may have or may not have the right to know about the *existence* of these documents. In our example from Fig. 9a, specific departments (car body engineering, electronic engineering, motor engineering) are responsible for generating expertise, i.e., three `generate expertise` activities are modeled in parallel. The different departments might or might not be allowed to access results of the other departments, or at least not before they generate their own expertises. These results may be considered as output data (e.g., Expertise documents) of the different `generate expertise` activities.[3]

Requirement 1.2 (AC rights definable on different objects of the model / instance levels). We define "object" as entity of a process model and process instance respectively; e.g., an *expertise document* produced as output of a `generate expertise` activity is considered as data object. The `generate expertise` activity itself as well as the change request (CR) process model are considered as two different objects. Moreover, a group of objects also

[2]The "days by person" measure is known in project management. Suppose we have one person working on a specific task, this measure specifies the number of days she needs in order to accomplish this task.

[3]A similar example stems from the "Articles Review" process. Consider reviewers of a specific article. Any of these reviewers is not allowed to access other reviewers' evaluation unless she finishes her own evaluation.

constitutes an object; e.g., AC rights may be defined 1) on all running CR process instances, or 2) on specific ones. We then define different levels of abstractions on objects. AC rights should be definable at these different levels of abstractions (cf. Requirement 3).

Example 5. For example, we need to distinguish between the AC rights defined on all change request (CR) process instances currently running and the AC rights defined on a specific CR process instance. An external partner may not have the right to access any of the running instances, while a CR initiator may not have the right to access specific process instances corresponding to change requests not initiated by her.

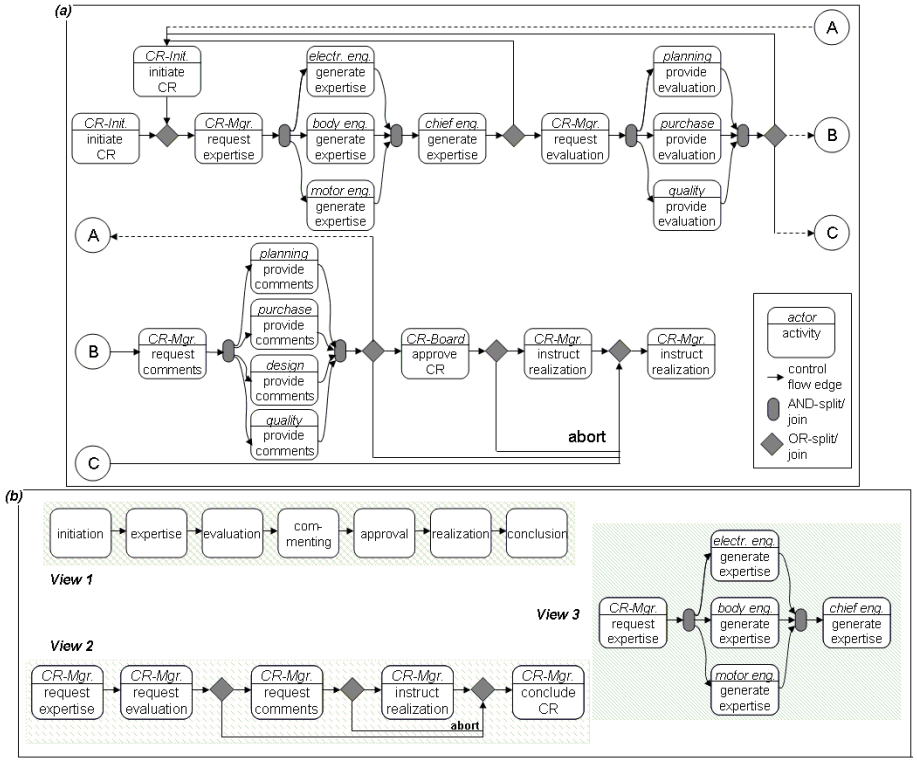

Figure 9: Automotive Domain – (a) Simplified Process of Dealing with Change Requests (CR), (b) Different Views on CR Process

Requirement 2 (Definition of static AC rights). We distinguish between "static" AC rights that are independent from the execution of a process instance, and "dynamic" AC rights for which this is not the case. The latter are based on elements such as activity status and control principles (e.g., separation of duties, dual control, and inter-case constraints) (Schaad and Moffett, 2002; Botha and Eloff, 2001).

Example 6. Regarding our CR process, a person from a specific department (e.g., motor engineering) responsible for generating expertise might not be allowed to access the *expertise document* generated by the other departments (e.g., car body engineering and electronic engineering) unless she finishes generating her own expertise. – This paper focuses on static access control rights.

Requirement 3 (Usability and maintainability of AC rights). AC rights should be simple to define and easy to maintain. As discussed in Tolone et al. (2005), a challenge is to balance collaboration and flexibility; i.e., we need to ensure that the advantages provided by process-aware IS are not reduced by AC rights being too rigidly defined. For this purpose, abstractions are required at the objects' level. In order to specify AC rights at different levels of granularity, we need to define hierarchies on objects;

Example 7. Regarding our CR scenario, it might be reasonable to authorize a manager to access all running CR process instances. However, regular users might only have access to specific CR instances (e.g., CR initiators only have the right to access CR process instances that correspond to change requests initiated by them).

Table 1 gathers major requirements identified. The ones highlighted (i.e., R1, R2, and R3) are addressed by the solution proposed in Section 5.

Table 1: *Access Control Major Requirements*

Requirements	Requirements' description
R1	**Definition of AC rights at a fine-grained level**
	R1.1 Meeting a spectrum of confidentiality
	R1.2 AC rights definable on diff. aspects of the mod./inst. levels
R2	**Definition of static AC rights**
R3	**Usability and maintainability of AC rights**
R4	Definition of dynamic AC rights
R5	Definition of AC rights on the visualization of a collection of processes
R6	Definition of AC rights for the look-ahead problem
R7	Completeness of the AC component

4 Candidate Solution Approaches for Access Control

Among a list of possible AC approaches, we feature two candidate solutions that we study and compare: the view- and the object-based approach. In both approaches we follow the main idea proposed by a generalized AC approach; i.e., RBAC (Role-Based Access Control) as described by Ferraiolo et al. (2001). In RBAC models AC rights are not directly linked to concrete users, but to roles.

The *view-based* approach consists of defining one basic view per user role; this view implicitly reflects the AC rights of the role over a process by only showing the information to be accessed by users with the respective role. The *object-based* approach, in turn, consists of defining, for each role, AC rights on the different aspects of a process (e.g., activity, activity attributes, process instance).

Section 4.1 illustrates the two featured approaches. Section 4.2 then summarizes their advantages and drawbacks. This helps us to clearly motivate the object-based approach as the one retained and elaborated in the following.

4.1 Description of Solution Approaches

View-based Approach. Considering a particular process model such as the CR process (cf. Fig. 9a), a number of views could be (manually) defined on this process. Each of them would then reflect the information accessible for users with a particular role. Access rights over the process may be derived implicitly from each view. Suppose the following views are defined on the CR process (cf. Fig. 9b):

1. *View 1.* High-level view on CR process,
2. *View 2.* View on `expertise` activities of CR process, and
3. *View 3.* View on `request` activities of CR process.

Then one basic view per role may be defined: *("general manager", View 1)*, *("CR manager", View 2)*, and *("engineer", View 3)*. Each of the views implicitly reflects the read access rights of the particular role:

- A *general manager* may access high-level activities like `initiation`, `expertise`, `evaluation`, `commenting`, and so on.
- *CR managers* may access activities `request expertise`, `request evaluation`, `request comments`, `instruct realization`, and `conclude CR`.
- *Engineers* may access concrete activities `request expertise` and `generate expertise`.

Object-based Approach. It consists of explicitly defining an extensible set of access rights for each role:

- *("general manager", {*`initiation`*,* `expertise`*,* `evaluation`*,* `commenting`*,* `approval`*,* `realization`*,* `conclusion`*}, Read)*
- *("CR manager", {*`request expertise`*,* `request evaluation`*,* `request comments`*,* `instruct realization`*,* `conclude CR`*}, Read)*

- *("engineer", {*`request expertise`*,* `generate expertise`*}, Read)*

A view may then be dynamically generated for a specific user based on the access rights associated with the role(s) played by this user. As an example, a view such as *View 3* illustrated in Fig. 9b would be generated for motor engineer *John Smith*.

4.2 Solution Approaches: Advantages and Drawbacks

We discuss the merits and shortcomings of these two approaches.

View-based Approach. The most obvious advantage comes from the fact that an existing concept (e.g., View Definition Language (Bobrik, 2008)) can be explicitly reused in order to reflect the access rights over processes. Hence, there is no need for defining a new AC language assuming that the process-aware IS clearly supports a View Definition Language). However, three drawbacks can be identified:

- *Costly maintenance of views:* Consider a process model P together with the views derived from it. Suppose a modification is brought to P: (1) the views affected by this change have to be identified possibly among a large number of existing views; (2) the identified views have to be adapted to reflect the change of P. This adaptation should be done without any failure; (3) the adapted views imply an implicit modification over AC rights.

- *Complexity of views combination:* Since a user may play more than one role (e.g., *John Smith* being a general manager as well as a motor engineer), we must be able to combine multiple views (e.g., *View 1* and *View 3*). The resulting view, automatically generated or manually modeled out of multiple views, will be shown to the user. On the one hand, we are facing a combinatorial problem (i.e., the different ways of arranging views in order to combine them). On the other hand, conflicts may exist between access rights reflected by the views to be combined. Such conflicts, first, must be detected, and second, be solved, probably by applying specific conflict resolution policies (di Vimercati et al., 2005; Jajodia et al., 2001) in order to correctly derive the combined view to be shown to the user.

- *Occurrence of redundant information due to lack of abstraction:* Suppose that a specific role R has access, among other things, to a specific activity A in all processes involving A. Using the view-based approach, this access right would be reflected by showing A within all the views respectively defined on the processes containing A. This leads to redundant information due to the definition of access rights at the level

of process models, not involving functional models (cf. Section 2.1). The redundancy of information is an issue not only for the view-based approach, but for other approaches as well, as long as the notion of abstraction is missing (e.g., at the level of activities). However, redundancy has more impact in conjunction with the view-based approach than in conjunction with the object-based one since for the latter the definition of abstractions is easier to achieve (cf. Section 5.3).

Object-based Approach. The main advantage of this approach is threefold. Indeed, the drawbacks identified for the view-based approach appear to be advantages here. First, there is no maintenance of views; the cost behind the maintenance operation is abolished. Second, views do not have to be combined and hence the complexity behind this operation does not exist. Third, if it is possible to define different levels of abstractions on objects, this will reduce redundancy when specifying access rights. The object-based approach may be criticized for not being intuitive since AC rights, instead of basic views, are initially defined for each role. However when compared with the drawbacks of the view-based approach, we voluntarily accept this only criticism, and select the object-based approach in order to elaborate the core solution for our logical AC model.

Table 2 summarizes the most important criteria that play either in favor of or against each of the considered approaches. As we can see, among five criteria, three play in favor of the object-based approach, while only one criterion plays in favor of the view-based approach.

Table 2: *Comparison of the View-based and Object-based Approaches*

Criteria/Approaches	**View-based**	**Object-based**
Ease of AC rights definition	+	-
Ease of AC rights maintenance	-	+
Ease of conflicts resolution	-	-
Ease of AC rights combination	-	+*
Redundancy-free	-	+

\+ Criterion plays in favor of the approach
\- Criterion plays against the approach
* This criterion is reduced to the "Ease of conflicts resolution" criterion

5 An Access Control Model

An AC model for process monitoring must allow to restrict access to authorized users only. Section 5.1 presents our formal framework for defining and manipulating AC rights. Section 5.2 and Section 5.3 discuss AC model extensions for coping with the problem of users playing multiple roles, and for addressing usability and maintainability issues.

5.1 Core AC Model

The specification of an AC module at the process monitoring level requires, first and foremost, the definition of access rights. A first step towards meeting *Req. R1* (cf. Table 1) consists of defining access rights on *attributes* associated with specific process aspects that we call *objects*.

Activities, process models or process instances are examples of accessed objects; attributes, indeed, reflect fine-grained characteristics of such objects. We first formally define the link between an object and its associated attributes; i.e., we define function *attributeSet* which determines all attributes associated with an object obj.

Definition 1 (Set of Attributes Associated with an Object) *Let ObjSet be the set of objects and AttSet be the set of attributes involved in the process monitoring component. Then: attributeSet: ObjSet* $\mapsto$ *AttSet*^{P} *with* $\forall att \in attributeSet(obj)$*: att is a valid attribute defined on obj.*

We associate with every object involved in the process monitoring component a set of attributes: $\forall obj \in ObjSet$: $attributeSet(obj) \subseteq AttSet$

Example 8. In order to illustrate Def. 1, we reconsider the process from Fig. 9a. For the sake of simplicity, we only retain the concrete concept of *activity* instead of the generalized one of *object*. Let ObjSet = {`request expertise`, `generate expertise`, `request evaluation`, `provide evaluation`, `request comments`, `provide comments`} be a set of activities involved in the CR process. Let further AttSet = {Att_1, Att_2, Att_3, Att_4, Att_5} be the set of attributes involved in the CR process. Taking into account Def. 1, suppose that the set of attributes associated with each activity is captured as follows: attributeSet(`req. expertise`) = {Att_1, Att_3}; attributeSet(`gen. expertise`) = {Att_1, Att_2, Att_4, Att_5}; attributeSet(`req. evaluation`) = {Att_1, Att_3}; attributeSet(`prov. evaluation`) = {Att_1, Att_2, Att_5}; attributeSet(`req. comments`) = {Att_1, Att_3}; attributeSet(`prov. comments`) = {Att_1, Att_2}. We may think of Att_1 as the *activity status* that could take values from the set {`NotActivated`, `Activated`, `Running`, `Completed`, `Skipped`}. Att_2 may be the *starting date/time* of an activity. Att_3 could be the *employee black list* with possible values {`Yes`, `No`} specifying whether this list should be taken into account (or not) when employees are chosen to work on a specific task (e.g., `generate expertise`). If this list is taken into account, employees on black list may be excluded from those that may work on the task.

Based on Def. 1, we retain two types of information that may be checked/-read: *the existence* and *the value* of an object's attribute. We distinguish between two different spectra of confidentiality defined on this information: 1) "Allow"/"don't allow" to check existence of an attribute within an object;

2) "Allow"/"don't allow" to read the value of an attribute within an object, or allow to read another form of the value. From this we derive Def. 2.

Definition 2 (Access Control on Existence/Value of Attribute) *Let (obj, att) (obj ∈ ObjSet, att ∈ attributeSet(obj)) denote an attribute att being associated with object obj. Then:*

$$Exist_{obj,att} := \begin{cases} 0 & if\ not\ allowed\ to\ check\ existence\ of\ att\ within\ obj \\ 1 & if\ allowed\ to\ check\ existence\ of\ att\ within\ obj \end{cases}$$

$$Val_{obj,att} := \begin{cases} 0 & if\ not\ allowed\ to\ read\ value\ of\ att\ within\ obj \\ 1 & if\ allowed\ to\ read\ only\ another\ form\ of\ value \\ 2 & if\ allowed\ to\ read\ value\ of\ att\ within\ obj \end{cases}$$

$Exist_{obj,att}$ determines whether or not it is allowed for someone to check for the existence of attribute *att* within object *obj*; $Val_{obj,att}$, in turn, determines whether or not it is allowed for someone to read the value of attribute *att* within object *obj*.

Example 9. Back to our example from Fig. 9a, suppose role "engineer" has the following access rights on the CR process: access to activities `request expertise` and `generate expertise`, access to the value of Att_1 and to another form of the value of Att_2, and access to the existence of Att_3 within `request expertise`. Taking into account Def. 2, the AC on the existence/-value of the different attributes can be captured as follows:

$Val_{\texttt{generate expertise},\ Att_1} = 2$, $Val_{\texttt{generate expertise},\ Att_2} = 1$,
$Val_{\texttt{request expertise},\ Att_1} = 2$, $Exist_{\texttt{request expertise},\ Att_3} = 1$

By default, we may suppose that the *closed policy*, considered as a classical approach for AC (see (Castano et al, 1995)), applies. If not specified otherwise:

$Val_{obj,att} = 0$ and $Exist_{obj,att} = 0$, $\forall$ obj $\in$ ObjSet, att $\in$ attributeSet(obj)

In this context, two classical approaches for AC are discussed by Castano et al (1995): *closed policy* where positive rights need to be specified explicitly, and *open policy* where negative rights need to be specified explicitly. The closed policy approach is known to ensure better protection than open policy. In the latter, the need for protection is not strong: by default, access is to be granted. Intuitively, we may also suppose that a specific operation prevails on another (cf. Fig. 10); e.g., whenever it is allowed to read the value of an attribute, this implies that it is also allowed to read another form of the value, and to check the existence of the attribute. Note that positive rights prevail on negative ones, i.e., positive rights are on bottom of the scale in

Fig. 10. This is because of the closed policy adopted.

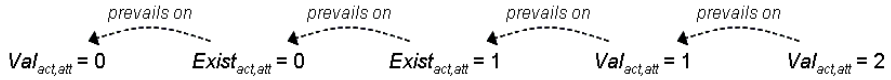

Figure 10: Prevailment of Access Rights

Example 10. Taking into account this scale, the following set of access rights is retained in the context of Example 9:

$Val_{\texttt{generate expertise},\ Att_1} = 2$, $Val_{\texttt{generate expertise},\ Att_2} = 1$,
$Val_{\texttt{request expertise},\ Att_1} = 2$, $Exist_{\texttt{request expertise},\ Att_3} = 1$,
$Exist_{\texttt{generate expertise},\ Att_4} = 0$, $Exist_{\texttt{generate expertise},\ Att_5} = 0$,
$Exist_{Activity,\ Attribute} = 0$, $\forall\ Activity \in \text{ObjSet} \setminus \{\texttt{request expertise}, \texttt{generate expertise}\}$, $Attribute \in \text{attributeSet}(Activity)$

Definition 3 (Attribute Value) *Let Dom_{AttSet} denote the value domain covering all potential values of attributes from AttSet. Then:*
Value: ObjSet $\times$ AttSet $\mapsto$ $Dom_{AttSet} \cup \{Undefined\}$ with Value(obj, att) either being the current value of attribute att on object obj or the value "Undefined" if att has not been written yet or att $\notin$ attributeSet(obj).

AC rights being clearly defined, we present now a mechanism consisting of two functions that respectively return (1) whether or not an attribute is associated with an object, (2) the exact value or an abstraction of the value of an attribute.

Definition 4 (Existence/Value of Attribute) *Let FunctionSet be the set of functions that can be applied on the value of an attribute in order to provide another form of this value. For setting the specific function that can be applied on a specific attribute, we define function fa: ObjSet $\times$ AttSet $\mapsto$ FunctionSet $\cup \{Undefined\}$ with fa(obj,att) mapping (obj, att) to a specific function from FunctionSet or to "Undefined" if att $\notin$ attributeSet(obj) or no function is defined. Then:*

f: ObjSet $\times$ AttSet $\mapsto$ AttSet $\cup \{Undefined\}$

$$with\ f(obj, att) := \begin{cases} att & if\ \ Exist_{obj,att} = 1 \wedge att \in attributeSet(obj) \\ Undefined & otherwise \end{cases}$$

h: ObjSet $\times$ AttSet $\mapsto$ $Dom_{AttSet} \cup Dom_{FunctionSet} \cup \{Undefined\}$

$$with\ h(obj, att) := \begin{cases} Undefined & if\ \ Val_{obj,att} = 0 \\ fa(obj, att)(Value(obj, att)) & if\ \ Val_{obj,att} = 1 \\ Value(obj, att) & if\ \ Val_{obj,att} = 2 \end{cases}$$

$$Dom_{AttSet} = \bigcup_{att \in AttSet} \quad Dom_{att}$$
$$Dom_{FunctionSet} = \bigcup_{fct \in FunctionSet} \quad Dom_{fct}$$

Basically, function f returns either the name of attribute att within object obj or "Undefined". Function h, in turn, determines either the value or another form of the value of attribute att within object obj, or "Undefined".

Example 11. If we go back to our example, applying Def. 4 would lead to the following existence / value of the different attributes:

$h(\texttt{generate expertise}, Att_1) = Value(\texttt{generate expertise}, Att_1)$
$f(\texttt{generate expertise}, Att_1) = Att_1$
$h(\texttt{generate expertise}, Att_2) = fa(\texttt{generate expertise}, Att_2)$
$\qquad\qquad\qquad\qquad\qquad\qquad (Value(\texttt{generate expertise}, Att_2))$
$f(\texttt{generate expertise}, Att_2) = Att_2$
$h(\texttt{request expertise}, Att_1) = Value(\texttt{request expertise}, Att_1)$
$f(\texttt{request expertise}, Att_1) = Att_1$
$h(\texttt{request expertise}, Att_3) = Undefined$
$f(\texttt{request expertise}, Att_3) = Att_3$
$h(\texttt{Activity}, Attribute) = f(\texttt{Activity}, Attribute) = Undefined$
for all other combinations of activities and attributes

The result of applying Def. 4 on our CR process, taking into account specific access rights assigned to role "engineer", is illustrated in Fig. 11.

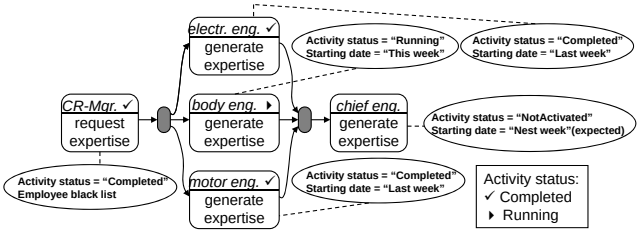

Figure 11: View on CR Process Provided to Role "Engineer"

5.2 Extended AC Model - Users Playing Multiple Roles

In this section, we recognize and point out the fact that a user may play more than one role leading to inconsistencies between the AC rights associated with each of the different roles.

Example 12. A user may play roles $r1$ = "manager" and $r2$ = "engineer" (cf. Fig. 12). On the one hand, engineers may not be given access to private information. On the other hand, managers may need to access private documents, and access to such information may be given to them.

In the given context, a number of conflict resolution policies are discussed in literature (di Vimercati et al., 2005; Jajodia et al., 2001; Fernandez et al., 1994; Shen and Dewan, 1992). None of them represents "the perfect solution". Whichever policy we take, we will always find one situation for which it does not fit. di Vimercati et al. (2005) states some problems of the different policies in conjunction with specific scenarios. Interestingly, conflicts may result either from explicitly defining negative AC rights, or from applying the closed policy. In the latter case, a simple solution approach may be to neglect negative AC rights derived from the used policy. Conflict resolution policies should be applied in the former case.

Example 13. Consider Process P from Fig. 12a) and its activities. Each of these activities is associated with a set of attributes for which AC rights need to be defined. Fig. 12c) depicts respective AC rights for roles $r1$ = "manager" and $r2$ = "engineer" respectively. For role $r1$ access to the values of all attributes of activities A, C and E shall be granted, while users with role $r2$ may access the values of attributes $Att1$ and $Att3$ of all activities. When applying Definition 2 we obtain the AC rights as depicted on the right hand side of Fig. 12c); a graphical illustration is given in Fig. 13a). While Fig. 13a) only depicts positive AC rights, Fig. 13b) implicitly adds negative ones as well. Assume now that a user u plays both roles $r1$ and $r2$. Then the question emerges what rights shall be granted to u. Regarding Fig. 13b) (with explicit positive AC rights and implicit negative AC rights), conflicts derive from the applied *closed policy*. In this simple scenario, they can be automatically handled by defining the set of AC rights for a user having roles $r1$ and $r2$ as the union of the two sets of positve rights (see Fig. 13c) However, conflicts may also derive from explicitly defining negative AC rights. As example, consider Fig. 13d): for role $r1$ a positive right to access $Att2$ of activity A exists, while for role $r2$ a negative right diallowing access to $Att2$ of activity A has been explicitly assigned. Consequently, a conflict exists for users having both roles. Then a conflict resolution policy needs to be applied (e.g., either permissions or denials taking precedence). – Due to lack of space, we abstain from further discussing this matter here and refer to existing literature on conflict resolution policies instead (di Vimercati et al., 2005; Jajodia et al., 2001; Fernandez et al., 1994).

5.3 Extended AC Model - Compact Definition of AC rights

So far, we have expressed that a certain attribute is allowed to be accessed (or not) within a certain object, particularly a certain activity. However, we must also be able to state within which processes this is allowed, i.e., what is the *context* of the AC to be defined. Candidates for the context are the entire process monitoring component (All), a group of process models, a particular process model, a group of process instances related to a particular

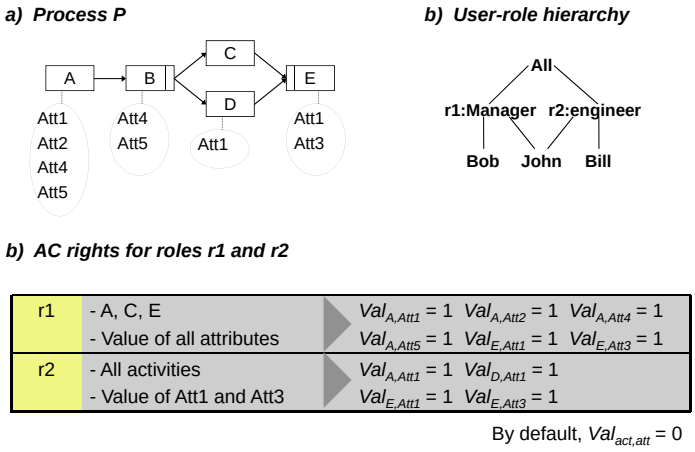

Figure 12: Granting Access Rights to User Roles

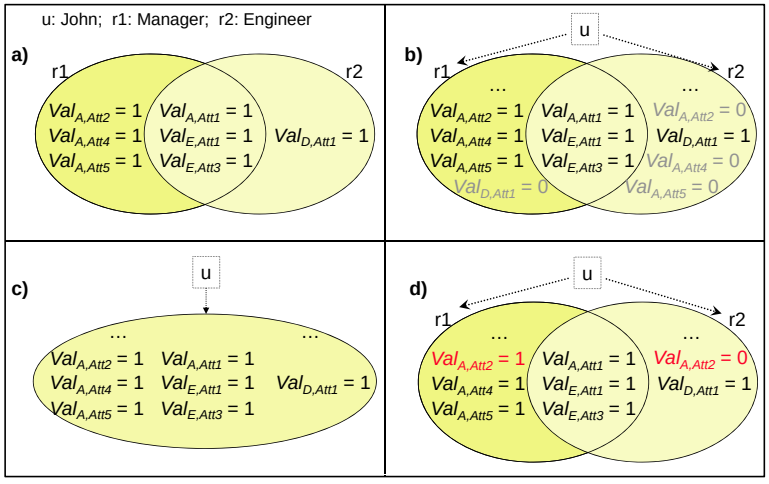

Figure 13: Possible Sets of Access Rights for a User with Two Roles

process model, and a process instance.

Example 14. The example elaborated in Section 5.1 presents a set of AC rights defined on a specific process model: CR_M. We may think of the following representation: (CR_M, $Val_{\texttt{generate expertise}, Att_1} = 2$) stating that the value of Att_1 from activity `generate expertise` is allowed to be read within process model CR_M. Suppose that AC rights are defined on a set of process models (e.g., M_1, M_2, M_3). This would lead to a set of couples: (M_1, $Val_{\texttt{generate expertise}, Att_1} = 2$), ($M_2$, $Val_{\texttt{generate expertise}, Att_1} = 2$), ($M_3$, $Val_{\texttt{generate expertise}, Att_1} = 2$).

When considering this example, we recognize the need for abstraction at the objects' level in order to compact the definition of AC rights reducing redundancy as much as possible. Therefore, one feasible way is to organize objects hierarchically (cf. Fig. 14): "All" at the top level, "Group of process models" at the next level down, "Process model" at the level just after, etc., and to propagate AC rights top-down. This allows us to meet the AC rights usability and maintainability requirement (cf. R3 in Table 1).

Example 15. Going back to our example, a group of process models $G_M = \{M_1, M_2, M_3\}$ would be defined, and the set of three couples would be reduced to the following couple: (G_M, $Val_{\texttt{generate expertise}, Att_1} = 2$). This approach would also simplify the definition of exceptions; e.g., it would be easy to express that no restrictions exist at all regarding accesses within any of the defined processes except the following: no accesses are allowed to activity `approve CR` within the CR process model. This would be reduced to: (All, $Val_{\texttt{All}, All} = 2$) (i.e., access is given to everything in order to bypass the closed policy), and (CR_M, $Exist_{\texttt{approve CR}, All} = 0$) (i.e., access is retrieved from `approve CR` within CR_M).

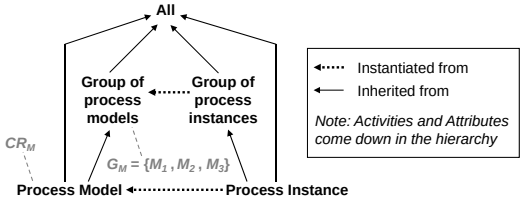

Figure 14: Objects' Hierarchy

6 Evaluation and Discussion

We have used the change mangagement process introduced at the beginning of this paper in order to illustrate the basic concepts of our AC model. We

have further evaluated the Proviado process monitoring framework and its access control model, respectively, in different case studies. The evaluation goal was to find out how well Proviado is suited for the monitoring of business processes whose data are scattered over distributed, heterogeneous information systems. In this section we focus on the evaluation of the Proviado AC model as suggested in this paper.

As sources for our evaluation we considered processes and process-aware information systems from the automotive domain as well as from healthcare. Regarding the processes from the automotive domain, we had access to 59 process models and could talk to process owners, process participants and IT departments. We first looked at electronic change management (ECM)[4] and at a supporting process-aware informations system. Furthermore, we investigated the processes dealing with car repair and car maintenance in garages, product release management, and product planning. With several hundred activities the product planning process was certainly the most complex one we considered. In particular, the implementation of this process was scattered over dozens of heterogeneous information systems and thus an integrated process monitoring component was a much needed module in this domain. As our second major data source we analyzed a process-aware clinical information system in the field of keyhole surgery. This system, which had been implemented using a commercial workflow engine an to which we had access, provided support for both administrative processes (e.g, patient admission or making appointments) and medical treatment processes (e.g., clinical diagnostics). Overall, it comprised 17 process models with up to 25 activities.

For all considered scenarios we identified relevant user roles in alignment with existing organizational models. We then analyzed the AC rights the different roles shall have in respect to the monitoring of processes and related data. This analysis was based on interviews with process owners and process participants on the one hand, and on a detailed analysis of the aforementioned information systems on the other hand. Following this, we tried to map the identified AC rights to our AC model as best as possible. In the following we will discuss the lessons learned from this.

First of all, the considered scenarios confirmed that the only feasible way to realize access control at the desired spectrum of confidentiality is to explicitly define the AC rights within the process monitoring component, hence getting rid of the burden to map access rights from the level of the information systems involved. We observed significant differences between the access control models applied in these information systems. Besides this, we identified additional user groups and roles respectively (e.g., clinical directors, business managers, system supervisors) that were particularly

[4]Regarding ECM standardized process models were published by the German Association of the Automotive Industry (VDA) (VDA, 2005).

interested in a process monitoring component, but were not directly involved in the operational processes.

When appyling the presented object-based AC model to the process monitoring scenarios, which we had identified in our case studies, we could define corresponding AC rights at the desired level of confidentiality and in a fine-grained way where required. In particular, we succeeded in using the described access control concepts to define static AC rights on different kinds of objects (like single activities, activity groups, single process instances, and so forth). Regarding the clinical processes, we additionally had to deal with cases in which users played multiple roles (e.g., a clinician working in different roles for different units in a university hospital). In most cases, we could restrict the specification of AC rights to positve rights and apply simple policies for conflict resolution. A more difficult task was to define AC rights in a compact and comprehensible manner. This was particularly challenging for large process models as in the case of product planning. Basically, the described context-based approach, which allows to specify AC rights in respect to a certain level within an objects' hierarchy, helped us to avoid an inflation of AC rights. More precisely, we applied the objects' hierarchy as depicted in Fig. 15 in our evaluation. Regarding the product planning process we explicitly defined additional views using the framework sketched in Section 2. We then assigned AC rights on the level of these views as well. – Overall, Requirements R1, R2 and R3 were satisifed by our AC approach.

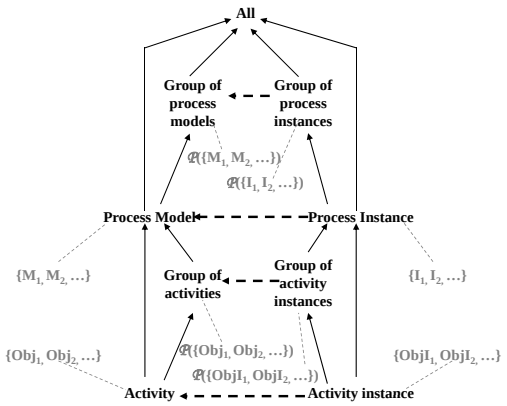

Figure 15: Objects' Hierarchy as Applied in Our Case Studies

Our case studies also revealed a number of limitations which will require further extensions of our AC approach. First of all, at the time we conducted our case studies, it was not possible to specify dynamic AC rights; e.g., constraints stating that a certain object may be only accessed by a user with a specific role and who was involved in the processing of this object before.

In this context, it was also not possible to make AC rights dependent on the state of a process instance; e.g., we could not express that a user may only access particular objects or object attributes, if the corresponding process instance (or relating process instances) has reached a particular state. Such dynamic or state-dependent AC rights were particularly relevant for the considered processes in electronic change management.

Another observation we made in the context of our clinical scenarios is that it is difficult to separate AC rights for process and application data. Ideally, access to the application objects (and their attributes) that are involved in the enactment of a particular process, should be tightly integrated with process execution and with access rights to the process. Only then, an integrated and harmonized definition of the AC rights on all aspects can be achieved. We believe that fundamental research on a tighter integration of object-aware and process-aware information systems is required in this context, rather than integrating all application objects into the monitoring component.

Finally, our current AC model showed limitations when being confronted with scenarios in which aggregated views were used (i.e., how to derive AC rights on abstracted visualizations) or in which a collection of process instances needs to be visualized in an integrated way. However, we do not see this as fundamental limitations for using our approach, but rather consider the concepts needed in this context as extensions of our approach.

7 Related Work

Similarly to the Proviado framework several other approaches target at the provision of appropriate process views and process visualizations for business performers. The techniques applied in this context include abstraction, aggregation, elimination, and modularization. Polyvyanyy et al. (2009, 2008) enable structural aggregations of the process logic in order to realize different levels of abstraction for business process models (e.g., by searching for meaningful process fragments suitable for generalization). Greco et al. (2005) propose an approach to process mining that combines process discovery strategies with abstraction methods with the aim of producing hierarchical views of the process that satisfactorily capture its behavior at different level of details. Therefore, at the highest level of detail, the mined model can support the design of executable workflows; at lower levels of detail, the views can be used in process execution platforms to support monitoring and analysis. Reijers et al. (2009) target at improved model management based on aggregated business process models. An extension of event-driven process chains is proposed, which can be used to describe a set of similar processes within one single model; i.e., the number of process models to be maintained is decreased. Like Proviado all these approaches enable abstract and aggregated

views on business processes. As opposed to Proviado, however, none of them deals with access control issues in connection with process monitoring and process visualization respectively.

The provision of adequate access control mechanisms is indispensable for any information system, and techniques like access control lists (ACL), capability lists and role-based access control (RBAC) have been proposed for dealing with respective security issues. In particular, RBAC models are widely used in existing information systems (Sandhu et al., 1996; Strembeck and Neumann, 2003; Wainer et al., 2003). Regarding process-aware information systems specific access control models have been proposed. Russell et al. (2005) describe various possibilities for assigning process activities to users. By contrast, permissions for accessing data and functions are mostly managed within invoked application systems. To deal with the latter problem Wu et al. (2002) suggest the concept of instance-based user groups. Each actor gets access to all data elements of the process instances in which he or she is involved; i.e., permissions to access data are assigned implicitly. However, such coarse-grained access to process instance data is not always acceptable in practice.

Wainer et al. (2003) suggest the W-RBAC access control model, which is based on a framework that couples an RBAC-based permission service and a process management component with clear separation of concerns. The permission service is based on an expressive logic-based language for selecting users that are authorized to perform certain process tasks. This basic model is further extended by incorporating exception handling capabilities through controlled and systematic overriding of security constraints. In Wainer et al. (2007) this security model is complemented by DW-RBAC, which additionally enables delegation and revocation of tasks in process management systems.

Over time, additional approaches for dealing in a secure way with specific issues related to process management were introduced. In the context of the ADEPT project (Reichert et al., 2003), for example, Weber et al. (2005) propose an extension to RBAC in order to support process changes safely. In the CEOSIS project, Rinderle and Reichert (2007) address changes that occur in respect to organizational structures. They discuss how to support such changes and how to correctly adapt access rules when the underlying organizational model is changed (Rinderle-Ma and Reichert, 2008, 2009).

To our best knowledge, none of the above approaches has addressed the problem of fine-grained AC in conjunction with process data integration and process monitoring yet (Muehlen, 2001; Junginger et al., 2004). This also applies in respect to existing process performance management tools (e.g., ARIS Process Performance Manager).

Some of the aspects retained in this paper have already been introduced by others. The fine-grained control was discussed by Tolone et al. (2005) as one of the collaborative environment factors that determine the usability of a specific AC model. The authors argue that it is not sufficient to define

AC rules only for groups of users on clusters of objects. A user might need a specific permission on an instance of an object at a particular point in time in the collaboration session. In Proviado, we were more explicit when defining AC rights at a fine-grained level: 1) we introduced the *spectrum of confidentiality* concept that would reflect the "specific" permission to grant or to revoke, and 2) we organize objects hierarchically such that AC rights may be defined in a *compact way* on the *different aspects* of the process model and instances. Strembeck and Neumann (2003) present another approach for fine-grained AC that uses special purpose RBAC constraints to base access control decisions on context information. Context dependencies are defined as dynamic RBAC constraints that check the actual values of contextual attributes for predefined conditions. If these conditions are met, the respective access request can be permitted. Accordingly, a conditional permission corresponds to an RBAC permission which is controlled by one or more context constraints. With this approach the advantages of RBAC are preserved on the one hand, while an additional means for defining and enforcing fine-grained context-dependent access control policies is provided on the other hand. Basically, such approach would be also beneficial in the context of business process monitoring.

Künzle and Reichert (2009) present a fine-grained AC model for data-driven processes. With this approach it becomes possible to restrict permissions to a selected set of process instances and corresponding object instances respectively. Thereby, restrictions can be defined depending on the relationships between users and object instances. Tolone et al. (2005), in turn, support permissions only being valid for a specific time space. This is an interesting point to be further investigated in Proviado as well. In the context of adaptive process-aware information systems, Weber et al. (2005) propose the definition of process type dependent AC rights . Only change commands that are useful within a particular *context* are allowed (e.g., activity *vacation request* must not be inserted in a CR process). This idea can be compared to our approach of specifying the context of an AC right. However, both approaches focus on different aims. Weber et al. (2005) further provides assistance for users when performing a change, whereas in this paper, the context notion is used for defining AC rights in a more focused way.

Sandhu and Thomas (1997) provide a task-based access control model that groups permissions for accessing data and functions. Whether or not a user may perform a particular task (e.g., process activity) depends on the agreement of another user at runtime. This makes it possible, for example, to manually approve access to process instances and their data. Similarly, a task-role based access control model (T-RBAC) is proposed by Oh and Park (2003). The authors classify tasks and consider them as fundamental unit of business work or business activity. T-RBAC deals with each task differently according to its class, and supports task level access control and

a supervision role hierarchy. As opposed to Proviado, however, with these approaches it is not possible to access data outside the scope of a specific task. Finally, Botha (2002) considers permissions for accessing data and functions in the context of an activity as well. This concept enables authorization for optional permissions in respect to data and takes the progress of a process instance into account as well. However, assignment of users to tasks 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 with same type.

8 Summary and Outlook

We presented the Proviado framework for process visualization and monitoring. In this context, we first introduced the Proviado process visualization approach. We then identified and discussed fundamental AC requirements. Following this we presented two possible solution approaches – the view-based and the objects-based approach – for major requirements, and we motivated the adavantages of the objects-based approach which we used for proposing a core AC model for business process monitoring.

We showed in detail how to describe positive and negative AC rights when using this AC model, and how prevailment of AC rights looks like in our approach. Further we discussed how to grant access to abstractions of object attribute values if required. Two extensions to this model were also presented. The first one deals with the problems that may appear when a single user plays more than one role; the second extension introduces the "context" notion and discusses the compact definition of AC rights taking into account a defined objects' hierarchy. Major requirements were addressed using the proposed AC model and its extensions. Finally, our evaluations have shown that our AC model allows to specify AC rights at the desired level of confidentiality and in a fine-grained way if required.

However, as discussed in Section 6 our evaluation also revealed additional challenges, e.g. regarding the management of dynamic AC rights and the tighter integration of process and data. We will address the discussed challenges in future work. In future research work will also include the investigation of advanced issues such as the aggregation and the definition of AC rights on data elements and other process aspects. Furthermore, in the PHILharmonic project, we are currently targeting at a close integration between process and data management. In this context, we consider access to processes and related objects in an integrated way taking into account dynamic aspects as well (e.g., the state of a process or object instance).

References

Bassil S, Reichert M, Bobrik R, Bauer T (2009) Access control for monitoring system-spanning business processes in Proviado. In: Proc. EMISA'09, pp 125–139

Bobrik R (2008) Konfigurierbare Visualisierung komplexer Prozessmodelle. PhD thesis, University of Ulm

Bobrik R, Reichert M, Bauer T (2005) Requirements for the visualization of system-spanning business processes. In: Proc. DEXA'05 Workshops, Copenhagen, pp 948–954

Bobrik R, Bauer T, Reichert M (2006) Proviado – personalized and configurable visualizations of business processes. In: Proc. EC-WEB'06, LNCS 4082, pp 61–71

Bobrik R, Reichert M, Bauer T (2007) View-based process visualization. In: Proc. BPM'07, LNCS 4714, pp 88–95

Botha R (2002) Cosawoe - a model for context-sensitive access control in workflow environments. PhD thesis, Rand Afrikaans University

Botha R, Eloff J (2001) Separation of duties for access control enforcement in workflow environments. IBM Systems Journal 40(3):666–682

Castano et al S (1995) Database Security. Addison Wesley

Costello C, Molloy O (2008) Towards a semantic framework for business activity monitoring and management. In: AAAI'08 Spring Symposium

Davis R (2008) ARIS Design Platform: Advanced Process Modelling and Administration. Springer

Fernandez E, Gudes E, Song H (1994) A model for evaluation and administration of security in object-oriented databases. IEEE ToKDE 6(2):275–292

Ferraiolo D, Sandhu R, Gavrila S, Kuhn D, Chandramouli R (2001) Proposed NIST standard for role-based access control. ACM ToISS 4(3):224–274

Greco G, Guzzo A, Pontieri L (2005) Mining hierarchies of models: From abstract views to concrete specifications. In: Proc. 3rd Int'l Conf. Business Process Management (BPM'05), pp 32–47

Groenewegen J, Hoppenbrouwers S, Proper E (2010) Playing archimate models. In: Enterprise, Business-Process and Information Systems Modeling. 11th Int. Workshop BPMDS'10 and 15th Int. Conference EMMSAD'10, Lecture Notes in Business Information Processing, vol 50, pp 182–194

Jajodia S, Samarati P, Sapino M, Subrahmanian V (2001) Flexible support for multiple access control policies. ACM ToDS 26(2):214–260

Junginger S, Huehn H, Bayer F, Karagiannis D (2004) Workflow-based business monitoring. In: Workflow Handbook 2004, Future Strategies, Lighthouse Point, pp 65–80

Kühn H, Bayer F, Junginger S, Karagiannis D (2003) Enterprise model integration. In: Proc. 4th Int. Conf. E-Commerce and Web Technologies (EC-Web'03)

Künzle V, Reichert M (2009) Integrating users in object-aware process management systems: Issues and challenges. In: Proc. Business Process Management Workshops 2009, Springer

Lenz R, Reichert M (2007) IT support for healthcare processes - premises, challenges, perspectives. Data and Knowledge Engineering 61(1):39–58

McGregor C (2002) The impact of business performance monitoring on WFMC standards. In: Fischer L (ed) Workflow Handbook'02, pp 51–64

McGregor C, Kumaran S (2002) Business process monitoring using web services in b2b e-commerce. In: Proc. Int. Parallel and Distributed Processing Symposium (IPDPS?02)

Müller D, Herbst J, Hammori M, Reichert M (2006) IT support for release management processes in the automotive industry. In: Proc. BPM'06, pp 368–377

Muehlen M (2001) Workflow-based process controlling. In: Workflow Handbook 2001, Future Strategies, Lighthouse Point

Mutschler B, Reichert M, Bumiller J (2008) Unleashing the effectiveness of process-oriented information systems: Problem analysis, critical success factors and implications. IEEE Transactions on Systems, Man, and Cybernetics 38(3):280–291

Oh S, Park S (2003) Task-role-based access control model. Information Systems 28(6):533–562

Polyvyanyy A, Smirnov S, Weske M (2008) Process model abstraction: A slider approach. In: Proc. EDOC'08, pp 325–331

Polyvyanyy A, Smirnov S, Weske M (2009) The triconnected abstraction of process models. In: Proc. BPM'09, pp 229–244

Reichert M, Dadam P, Bauer T (2003) Dealing with forward and backward jumps in workflow management systems. Software and Systems Modeling 2(1):37–58

Reijers H, Mans R, van der Toorn R (2009) Improved model management with aggregated business process models. Data and Knowledge Engineering 68(2):221–243

Rinderle S, Reichert M (2005) On the controlled evolution of access rules in cooperative information systems. In: Proc. 13th Int'l Conf. on Cooperative Information Systems (CoopIS'05), pp 238–255

Rinderle S, Reichert M (2007) A formal framework for adaptive access control models. In: Journal of Data Semantics, IX, LNCS 4601, pp 82–112

Rinderle S, Bobrik R, Reichert M, Bauer T (2006) Businesss process visualization - use cases, challenges, solutions. In: Proc. 8th Int'l Conf. on Enterprise Information Systems (ICEIS'06), Track on Information Systems Analysis and Specification, pp 204–211

Rinderle-Ma S, Reichert M (2008) Managing the life cycle of access rules in CEOSIS. In: Proc. EDOC'08, pp 257–266

Rinderle-Ma S, Reichert M (2009) Comprehensive life cycle support for access rules in information systems: The CEOSIS project. Enterprise Information Systems 3(3):219–251

Russell N, van der Aalst W, ter Hofstede A, Edmond D (2005) Workflow resource patterns: Identification, representation and tool support. In: CAiSE'05, pp 216–232

Sandhu R, Thomas R (1997) Task-based authorization controls (TBAC): A family of models for active and enterprise-oriented authorization management. In: Proc. IFIP'97, pp 166–181

Sandhu R, Coyne E, Feinstein H, Youman C (1996) Role-based access control models. IEEE Computer 29(2):38–47

Schaad A, Moffett J (2002) A framework for organisational control principles. In: Proc. ACSAC'02, Las Vegas, pp 229–238

Shen H, Dewan P (1992) Access control for collaborative environments. In: Proc. CSCW'92, pp 51–58

Strembeck M, Neumann G (2003) An integrated approach to engineer and enforce context constraints in rbac environments. ACM Trans Inf Syst Secur 7(3):392–427

Tolone W, Ahn GJ, Pai T (2005) Access control in collaborative systems. ACM Computing Surveys 37(1):29–41

VDA (2005) German association of the automotive industry, engineering change management. part 1: V 1.1., doc. no. 4965

di Vimercati SDC, Samarati P, Jajodia S (2005) Policies, models, and languages for access control. In: Proc. Int'l Workshop DNIS'05, Aizu-Wakamatsu, pp 225–237

Wainer J, Barthelmess P, Kumar A (2003) W-rbac - a workflow security model incorporating controlled overriding of constraints. Int J Cooperative Inf Syst 12(4):455–485

Wainer J, Kumar A, Barthelmess P (2007) DW-RBAC: A formal security model of delegation and revocation in workflow systems. Information Systems 32(3):365–384

Weber B, Reichert M, Wild W, Rinderle S (2005) Balancing flexibility and security in adaptive process management systems. In: CoopIS'05, LNCS 3760, pp 59–76

Weske M (2007) Business Process Management: Concepts, Languages, Architectures. Springer

Wu S, Sheth A, Miller J, Luo Z (2002) Authorization and access control of application data in workflow-systems. Journal of Intelligent Information Systems 18:71–94