Towards Object-aware Process Management Systems: Issues, Challenges, Benefits

Vera Künzle[1] and Manfred Reichert[1]

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

Abstract. Contemporary workflow management systems (WfMS) offer promising perspectives in respect to comprehensive lifecycle support of business processes. However, there still exist numerous business applications with hard-coded process logic. Respective application software is both complex to design and costly to maintain. One major reason for the absence of workflow technology in these applications is the fact that many processes are data-driven; i.e., progress of process instances depends on value changes of data objects. Thus business processes and business data cannot be treated independently from each other, and business process models have to be compliant with the underlying data structure. This paper presents characteristic properties of data-oriented business software, which we gathered in several case studies, and it elaborates to what degree existing WfMS are able to provide the needed object-awareness. We show that the activity-centered paradigm of existing WfMS is too inflexible in this context, and we discuss major requirements needed to enable object-awareness in processes management systems.

Key words: Workflow Management, Object-aware Process Management Systems, Data-driven Process Execution

1 Introduction

Nowadays, specific application software (e.g., ERP, CRM, and SCM systems) exists for almost every business division. Typically, respective software enables access to business data and offers a variety of business functions to its users. In addition, it often provides an integrated view on the business processes. Though such tight integration of process, function and data is needed in many domains, current application software still suffers from one big drawback; i.e., the hard-coding of the process and business logic within the application. Thus, even simple process changes require costly code adaptations and high efforts for testing. Existing application software typically provides simple configuration facilities; i.e., based on some settings one can configure a particular process variant. Problems emerging in this context are the lack of transparency of the configurable processes and the mutual dependencies that exist between the different configuration settings. In addition, like the overall process logic the settings are often (redundantly) scattered over the whole application code, which therefore

becomes complex and difficult to maintain over time. This results in long development cycles and high maintenance costs (e.g., when introducing new features).

In principle, *workflow management systems* (WfMS) offer promising perspectives to cope with these challenges. Basically, a WfMS provides generic functions for modeling and executing processes independent from a specific application. Contemporary WfMS, however, are not broadly used for realizing data- and process-oriented application software, particularly if a close integration of the process and the data perspective is needed. In the latter case the processes are typically data-driven; i.e., the progress of single process instances does not directly depend on the execution of activities, but on changes of attribute values of data objects. Thus business processes and data cannot be treated independently from each other, and business process models need to be compliant with the underlying data structure; i.e. with the life cycles of the used data objects.

In this paper we demonstrate why the activity-centered paradigm of existing WfMS is inadequate for supporting data-oriented processes. For this purpose, we elaborate important properties of existing application software and show to what degree they can be covered by existing WfMS. Based on the identified shortcomings, we define major requirements for a generic system component enabling*data-oriented processes* with integrated view on both business processes and business data. To clearly distinguish this approach from existing WfMS we denote it as *Object-aware Process Management System* in the following.

Section 2 summarizes characteristics of contemporary WfMS and introduces an example of a data-oriented process. We use this running example throughout the paper to illustrate different issues relevant for the support of data-oriented processes. In Section 3 we describe five key challenges for realizing an Object-aware Process Management System. We check to what degree contemporary WfMS cover the properties of data-oriented applications. Based on the problems identified in this context we derive the requirements for Object-aware Process Management Systems. Section 4 describes related work. The paper concludes with an outlook on our future research in Section 5.

2 Backgrounds and Illustrating Example

This section describes basic workflow terminology and introduces an illustrating example. Based on this information we discuss the deficiencies of contemporary WfMS in the following sections.

Existing WfMS. In existing WfMS, a process definition consists of a set of activities and their control flow [1]. The latter sets out the order and constraints for executing the activities. It can be defined based on a number of workflow patterns which, for example, allow to express sequential, alternative and parallel routing as well as loop backs [2]. Each activity, in turn, represents a particular task and is linked to a specific function of an application service. To be able to assign human tasks to the respective actors, in addition, actor expressions (e.g., user roles) need to be defined for the corresponding activities. 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 activities preceding in the control flow are completed or cannot be executed anymore (except loop backs). When an interactive activity becomes enabled, corresponding work items are added to the work lists of responsible users. Finally, when a work item is selected by a user, the WfMS launches the associated application service.

Example of a data-oriented process. We consider the (simplified) process of a job application as it can be found in the area of human resource management. Using an online form on the Internet, interested candidates may apply for a vacancy. The overall goal of the process then is to decide which applicant shall get the offered job. A personnel officer 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 candidate shall be invited for an interview), and submit their recommendation to the personnel officer. Based on the provided reviews the personnel officer makes his decision on the application(s) or he initiates further steps like an interview or another review. In general, different reviews may be requested and submitted respectively at different points in time. In any case, the personnel officer should be able to sign already submitted reviews at any point in time.

3 Findings, Problems, Requirements

In several case studies we have evaluated the properties of data- and process-oriented application software. This section summarizes basic findings from these studies and illustrates them along our running example. We then reveal characteristic problems that occur when using existing workflow technology for implementing the identified properties. This leads us to a number of fundamental requirements to be met by *object-aware process management systems*.

3.1 Challenge 1: Integration of Data

Findings. Usually, application systems manage data in terms of different *object types* represented by a set of *attributes*. At runtime, for each object type several *object instances* exist, which differ in the values of their attributes. Each object type has at least one attribute representing its *object ID*. Using this attribute any object instance can be uniquely identified. Relationships between object types are described using attributes as well. At runtime, object IDs of other object instances are then assigned to these attributes. Generally, an object instance may be referenced by multiple other object instances of a particular object type.

Business Data is represented by a variable number of object instances of different object types which are related to each other.

Fig. 1a depicts the data structure for our running example. For each application multiple reviews can be requested (cf. Fig. 1b). Thereby the precise number

of related object instances varies from case to case; i.e., the number of requested reviews may differ from application to application, and it may also change during runtime (e.g., if for an application some reviews are requested or completed later than others).

In data- and process-oriented applications, available information can be accessed by authorized users at any point in time regardless of the process status.

From a user perspective, the instances of a particular object type correspond to rows in a table. Table columns, in turn, relate to selected attributes of the object type or – more precisely – to attribute values of the object instances. Attributes representing object relationships are resolved; i.e., their values are substituted by (meaningful) attribute values of the related object instances. Additional information on object instances (e.g., attributes not displayed by default within the table or detailed information about referenced object instances) can be viewed on-demand. Using this data- or object-centric view, besides working on *mandatory process activities* , authorized users may optionally edit attribute values of single object instances at arbitrary points in time (*optional activities*).

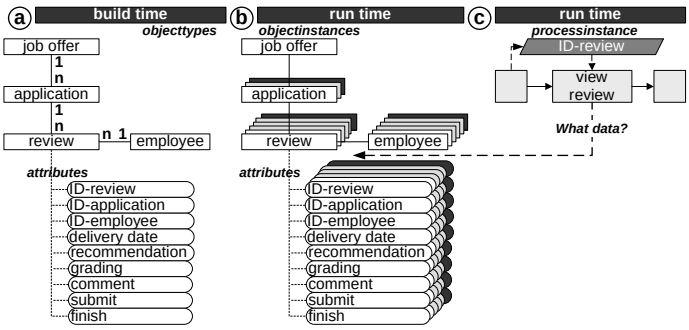

Fig. 1. Data structure and access to context information

Problems. Existing WfMS are unable to provide such object-centric views. Most of them only cover simple data elements, which store values of selected object attributes, while the object instances themselves are stored in external databases. More precisely, only the data needed for control flow routing and for supplying input parameters of activities are maintained within the WfMS (i.e., so-called workflow relevant data), while other application data is unknown to it. Obviously, this missing link between application data and business process prohibits an integrated access to them; i.e., access to detailed business information is only possible when executing an activity and its related application function respectively. Fig. 1c shows a process activity for perusing a particular review. Which review shall be displayed can be controlled by the WfMS by handing over its objectID to the invoked activity. However, the WfMS cannot control which attributes of the review object or of related objects (e.g., the application) can be accessed. Missing or incomplete context information, however, often leads to inefficient work and erroneous results [3].

In principle, optional activities, enabling access to application data at arbitrary points in time, could be emulated in WfMS by explicitly modeling them at different positions in the control flow. However, this would lead to spaghetti-like process models with high number of redundant activities, which are difficult to comprehend for users. Besides this, users would not be able to distinguish optional activities from mandatory ones. Fig. 2 illustrates this problem along our running example. Here, optional activity `edit data` is embedded multiple times in the process definition in order to be able to access application data if required. Note that without such an explicit integration of optional activities, needed changes of application data would have to be accomplished directly within the applications system. When bypassing either the WfMS or appl. system, however, inconsistencies with respect to attributes, redundantly maintained in both systems, might occur. Worst case, this can result in runtime errors or faulty process executions.

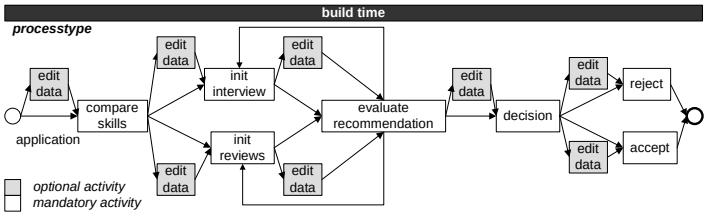

Fig. 2. Mandatory and optional activities in contemporary WfMS

Requirements. Object-aware process management systems need to be tightly integrated with application data. In particular, these data should be manageable and accessible based on complex objects rather than on atomic data elements. Another challenge is to cope with the varying and dynamic number of object instances to be handled at runtime. Thereby, the different relations between the object instances have to be considered as well. Finally, regardless of process status, it should be possible to access object information at any time.

3.2 Challenge 2: Choosing granularities for processes and activities

Findings. *For different object types separate process definitions exist [4]. The creation of a process instance is directly coupled with the creation of an object instance; i.e., for each object instance exactly one process instance exists.*

Fig. 3 illustrates the mapping between object and process types as well as between object and process instances. The object type of a job application has its own process type. At runtime, there are several instances of a job application object. Correspondingly, for each of them a separate process instance is created.

Regarding the process type associated with a particular object type, each activity refers to one or more attributes of the object type. There is one action per attribute to read or write its value. Each activity consists of at least one action.

When executing a particular process instance related subordinate processes may be triggered. Results collected during their execution are relevant for the

execution of the superordinate process instance as well. In this context the creation of a subordinate process instance is also coupled with the creation of a corresponding object instance. The latter has to refer to the object instance of the superordinate process instance. Consequently, the number of subordinate process instances depends on the number of object instances which reference the object instance associated with the superordinate process instance.

The relations between process types correspond to the relations between object types within the overall data structure [4].

Fig. 3 illustrates the analogy between data structure and process structure. For each job application an arbitrary number of reviews may be requested, and for each review object one process instance is running. The latter constitutes a subordinate process of the process instance related to the job application.

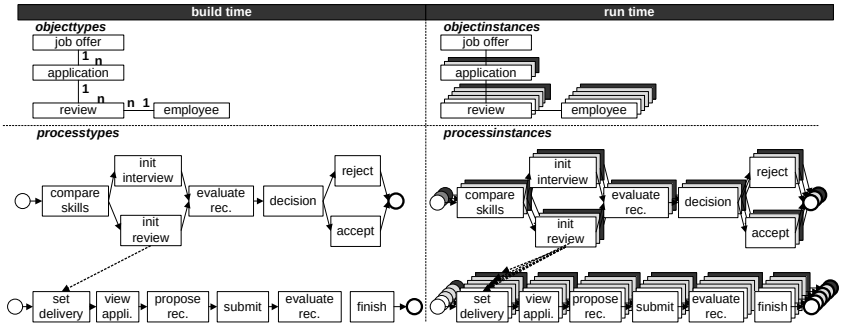

Fig. 3. Analogy between data and process structure

Problems. Granularity issues are not adequately addressed in existing WfMS; i.e., processes, sub-processes and activities may be modeled at arbitrary level of granularity. Neither a uniform methodology nor practical guidelines exist for process modeling [5], often resulting in inconsistent or non-comparable models. Furthermore, when modeling and executing processes in WfMS, there exists no direct support for considering the underlying data structure; i.e., the objects and their relations. In particular, two drawbacks can be observed: First, the creation of (sub) process instances cannot be coupled with the creation of object instances. Second, in many WfMS the number of sub process instances has to be fixed already at build time [6]. Note that WfMS enabling multi-instance patterns constitute an exception in the latter respect [2].

Requirements. The modeling of processes and data constitute two sides of the same coin and therefore should correspond to each other [5]. Thereby, we have to distinguish between object level and (data) structure level: First, a process type should always be modeled with respect to a specific object type; process activities then may directly relate to attributes of this object type. Second, at the structure level, process relations should correspond to the ones between the corresponding data objects. Finally, instantiation of processes needs to be coupled with the creation of related object instances.

3.3 Challenge 3: Data-based Modeling

Findings. *The progress of a process instance correlates with the attribute values of the associated object instance. Corresponding to this, the steps of a process are less defined on basis of black-box activities, but more on explicit data conditions.*

Fig. 4 shows an instance of a review object together with the related process instance. For each process step, pre-conditions on the attribute values of the object instance as well as the attribute values changed within this step are depicted. In particular, the process is defined by setting goals described in terms of conditions on object attribute values. Regarding our example from Fig. 4, these data conditions are related to the attributes of the review object. This way, process state and object state sync at any point in time. Mandatory activities can be identified by analyzing the data conditions. More precisely, they comprise those actions changing object attributes in a way such that the conditions for executing subsequent activities become fulfilled [3]. For each process step at least one mandatory activity exists.

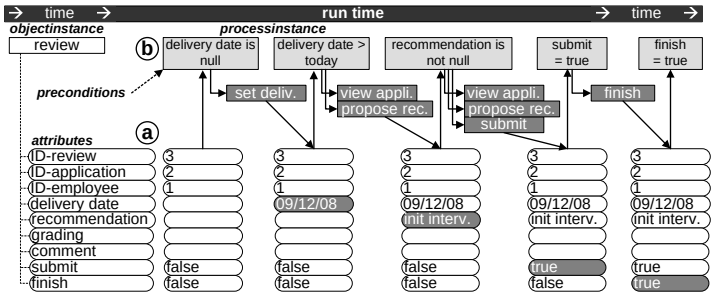

Fig. 4. Progress within data and data-based modeling

Problems. In existing WfMS, process designers have to explicitly define the activities to be carried out as well as their order constraints. In particular, no support exists for verifying whether or not the (semantic) goals of a process can be achieved [7, 8, 9]. Some approaches define pre- and post-conditions for certain activities in relation to application data. If the pre-conditions of such an activity cannot be met during runtime, however, process execution is blocked. In this context, it is no longer sufficient to only postulate certain attribute values for executing a particular activity. It then must be also possible, to dynamically react on current attribute values.

Requirements. In object-aware process management systems, the modeling of a process type should not be based on the activities to be carried out. Instead, process steps should be defined in terms of data conditions. The latter, in turn, should relate to the attributes belonging to the corresponding object type.

3.4 Challenge 4: Synchronizing Process Instances

Findings. A subordinate process is always instantiated during the execution of another process instance [6]. Like for the superordinate process instance, a

corresponding object instance is then created. In particular, this object instance references the one related to the superordinate process instance. Finally, the pre-condition of the process step, in which the subordinate process instance is created, corresponds to a data condition on the superordinate object instance.

The creation of a particular object instance depends on the progress of the process instance related to the superordinate object instance.

Fig. 5a illustrates this relationship. A new review object cannot be created before the skills of the applicant have been compared with the job profile.

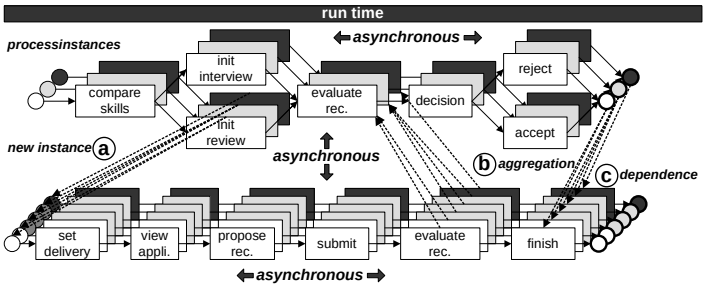

Fig. 5. Synchronizing process instances

During the execution of a superordinate process instance, information from its subordinate process instances may be used for decissions within the superordinate process instance.

To accomplish such evaluation, data of multiple subordinate object instances may be required [6]; i.e., we need to aggregate the values of particular attributes of subordinate object instances. Which subordinate object instances shall be taken into account may depend on the execution progress of their corresponding process instances. Fig. 5b illustrates this along our running example. Within the parental process instance handling a particular job application, the requested reviews (i.e., results from different subordinate processes) are jointly evaluated. Thereby, only submitted reviews are considered.

The executions of different process instances may be mutually dependent [4, 6]. Respective dependencies may exist between instances of the same process type as well as between instances of different process type.

Considering this, the data conditions for executing process steps are even more complex in existing application software than described above; i.e., these data conditions may be not only based on the attributes of the corresponding object type, but also on the attributes of related object types. For example, a review may only be marked as `completed` after a decision on the job application has been made (cf. Fig. 5c).

Problems. In existing WfMS, process instances are executed in isolation to each other [6]. Neither dependencies between instances of different process types nor dependencies between instances of the same process type can be defined at a reasonable semantical level. Often, the modeling of subordinate processes serves as a workaround. However, in existing WfMS the execution of subordinate process instances is tightly synchronized with their superordinate process instance;

i.e., the latter is blocked until the sub process instances are completed. Thus, neither aggregated activities nor more complex synchronization dependencies as described above can be adequately handled in WfMS [6].

Requirements. Generally, it should be possible to execute both instances of the same and instances of different process types in a loosely coupled manner, i.e., 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. Furthermore, to a superordinate process instance several subordinate process instances should be assignable in accordance with the relationships between corresponding object instances as well as their cardinalities.

3.5 Challenge 5: Flexibility

Findings. As described, there are optional as well as mandatory activities. The former are used to gather object information at any point in time regardless from the progress of the corresponding process instance. Opposed to this, the latter are mandatory and comprise actions that change the values of the object attributes used within the data conditions of one or multiple process steps.

The activation of an activity does not directly depend on the completion of other activities; i.e., it may be executed as soon as its data condition is satisfied. An activity can be also executed repeatedly as long as its data condition is met. Depending on how the data conditions of the different activities look like, a more or less asynchronous execution becomes possible (cf. Fig. 6).

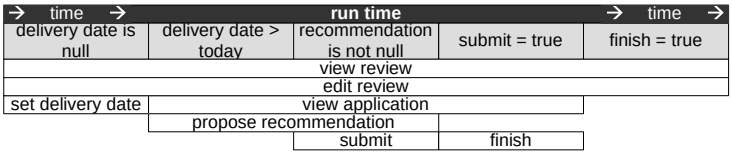

Fig. 6. Asynchronous and overlapping execution of activities

Generally, activities consist of one or more atomic actions for reading or writing the different attributes of an object instance. Which object attributes can be actually modified in a given context depends on the progress of the related process instance. For example, Fig. 7 shows the different actions available within the (optional) activity for entering the data of a review object. As can be seen, the concrete set of selectable actions depends on the conditions actually met by the object instance; i.e., (optional) activities dynamically adapt their behavior to the progress of the corresponding process instance (denoted as *horizontal dynamic granularity*). Interestingly, the attribute changes required to fulfill the data condition of a particular process step can be also realized when executing an optional activity. Since this can be done asynchronously at arbitrary point in time, high process flexibility can be achieved. Furthermore, for a particular activity optional and mandatory actions can be differentiated. Fig. 7 shows the mandatory actions for a review. Note that these actions may differ from step to step. As opposed to optional activities, mandatory ones only include those actions necessary for the fulfillment of the data conditions of subsequent steps.

Mandatory activities belonging to different instances of the same process type may be executed together.

Required data is only entered once by the user; i.e., users may group a number of activities for which they want to provide the same input data (denoted as *vertical dynamic granularity*). Fig. 8 illustrates this for activity `finish`.

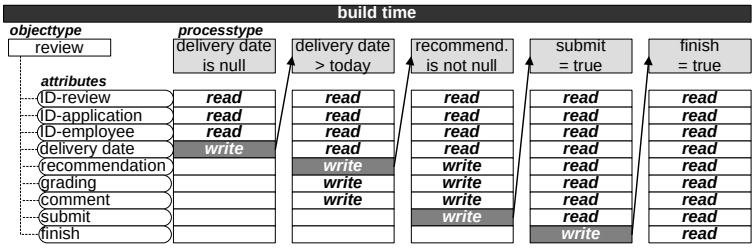

Fig. 7. Granularity of activities with optional and mandatory actions

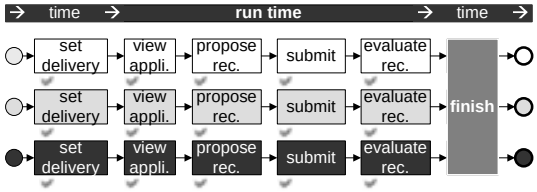

Fig. 8. Grouping of activities

Problems. Due to the activity-driven execution of process instances in existing WfMS, an activity can usually be activated only once (except loop backs). Furthermore, activity execution must take place in a precisely defined context. However, such rigid behavior is not always adequate. Sometimes an activity needs to be repeated spontaneously; or it has to be executed in advance, or first be stopped and then be caught up at a later point in time [3]. Conventional WfMS do not allow for this kind of flexibility. Furthermore, users are typically involved in the execution of multiple instances of a particular process type. Thus, their worklist usually contains many activities of same type. However, each of them needs to be processed separately in WfMS, which does not always comply with common work practice. In summary, the isolated execution of process instances in existing WfMS is too inflexible [10].

Requirements. Data-driven process execution is needed; i.e., process execution should be not guided by activities, but rather be based on the state of the processed object instances. Thereby, a much more flexible execution behavior and optional activities can be realized. Furthermore, it should be possible to make the selectable actions within an activity execution dependable on the state of the process instances. Finally, it should be possible to work on several activities with same type, but belonging to different process instances, in one go.

The above discussions have revealed the limitations of current WfMS. Only

being able to cope with atomic or stateless data elements is by far not sufficient. Instead, tight process and data integration is needed. This integration can based on objects, object attributes and object relations. Therefore, these three levels need to be reflected in process definitions as well; i.e., activities should be related to object attributes and process modeling should be based on objects. The hierarchical relations between processes and other process interdependencies then depend on object relations; i.e., on references between objects. In summary, we need comprehensive support for the data-based modeling and data-driven execution of business processes.

4 Related Work

The described challenges have been partially addressed by existing work. However, a comprehensive solution for object-aware process management is still missing. Fig. 9 summarizes what challenges have been addressed by which approach.

		Artifact Centric Modelling *IBM Research USA*	Production Based Support *University of Eindhoven*	Data Driven Coordination *University of Twente / Ulm*	Case Handling *University of Eindhoven, Hasso Plattner Institute Potsdam*	Proclets *University of Eindhoven, Colorado, Campinas*	Batch activities *University of Queensland*
integration of data	atomic elements / attributes	x	x		x	x	x
	objects	x		x	o		
	relations between data	x	x	x			
	flexible quantity			o			
	access beyond activities				x		
granularity	activity	x	x		x		
	process		o	x	o	x	
databased modelling		o	x	o	x		
synchronisation				x		x	
flexibility	horizontal dynamic granul.				x		
	vertical dynamic granularity						x
	data-driven execution		x	o	x		

Fig. 9. Overview of related work

Challenge 1: Integration of Data Concepts for better integration of processes and data are suggested in Artifact-Centric Modeling [11], Production-Based Workflow Support [5, 12], Data-Driven Process Coordination (Corepro) [4], and Case Handling [3]. [12] establishes relations between atomic data elements, but neither supports complex objects nor varying numbers of data elements. Corepro, in turn, allows to model objects and object relations [4, 13]; object definition does not consider attributes and cardinalities of object relations. In [11], so-called *artifacts* have to be identified first. Like objects, artifacts consist of different attributes which can be also used to define relations between them. Unlike objects, they are not defined at type level and therefore cannot be instantiated multiple times. In all approaches, access to data is only possible in the context of an activity execution, i.e. at a certain point during process execution. Only *Case Handling* [3] allows access to data outside the scope of activities, but does not provide explicit support for complex objects and data structures.

Challenge 2: Choice of Granularity for Activities and Processes. Objects and object relations constitute guidelines for choosing the granularity for processes, sub-processes and activities. Process definitions are based on ob-

jects and activities are used to modify the values of corresponding attributes. Furthermore, a process structure should be in accordance with the data structure. Granularity issues are addressed by the previously mentioned approaches and by Proclets [6]. However, none of them enables complete process definition with references to attributes, objects and object relations. In [11] activities are modeled based on one ore more artifacts, but without deriving the granularity of processes automatically. Proclets [6] are lightweight processes, which communicate with each other via messages. The granularity of a process is not explicitly defined. By considering the quantity of process instances, an implicit analogy between processes and objects can be drawn. Corepro [4] explicitly coordinates individual processes based on the underlying data structure. The granularity of processes and activities can be chosen freely. [5, 12] consider both the granularity of activities and the one of processes. Activities always relate to one or more atomic data elements. The structure of the process corresponds to the relationships between the data elements. Sub-processes do not exist. In [3] activities are described in terms of atomic data elements as well. Due to their indirect encapsulation, a process is defined based on an individual "case". However, relationships are not considered.

Challenge 3: Data-based modeling. Though [11] does not allow for data-based modeling, activities are defined with references to the identified artifacts. In Corepro, process coordination is realized in accordance with the objects and their relations. Objects are defined in terms of states and transitions between them. Furthermore, processes assigned to different objects can be related to each other based on external transitions. The most advanced approaches in relation to data-based modeling are provided by [3] and [5, 12]. Data-based modeling of activities in terms of atomic data elements is possible. However, for each process step still an activity has to be explicitly defined.

Challenge 4: Synchronization. In [6], processes are synchronized based on messages. Thereby, a variable number of process instances is considered. However, their synchronization is not explicitly based on the data structure. The most powerful approach in the given context is provided by the data-driven coordination of processes in Corepro [4]. Process synchronization is in accordance with the related data structure. Thereby, a variable number of instances can be created. The creation of new object instances at runtime is possible, but requires an ad-hoc change of the related data structure [13].

Challenge 5: Flexibility. Case Handling [3] enables *horizontal dynamic granularity*. A data element can be read and written within several activities. These data elements can either be free, mandatory or optional. A data element which is mandatory for an activity can be optional for preceding ones. [10] enables *vertical dynamic granularity* of activities; same activities of different process instances can be grouped and executed together. [3, 12] enable the data-driven execution of processes based on current values of the data elements. In Corepro [4] processes themselves are still activity-driven, whereas process synchronization follows a data-driven approach.

5 Outlook

Our overall vision is to develop a framework for object-aware process management; i.e., a generic component for enabling data-driven processes as well as an integrated view on process and data. On the one hand we want to provide similar features as can be found in some hard-coded, data-oriented applications. On the other hand we want to benefit from the advantages known from workflow technology. However, a tight integration of data and process is only one of the challenges to be tackled. Other ones arise from the involvement of users and the handling of access privileges; e.g., depending on object data. 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. Aalst, W., Hee, K.: Workflow-Management - Models, Methods and Systems. MIT Press, Cambridge, MA, USA (2004)
2. Aalst, W., Hofstede, A., Kiepuszewski, B., Barros, A.: Workflow patterns. Distr. & Parallel Databases **14** (2003) 5–51
3. Aalst, W., Weske, M., Grünbauer, D.: Case handling: A new paradigm for business process support. DKE **53**(2) (2005) 129–162
4. Müller, D., Reichert, M., Herbst, J.: Data-driven modeling and coordination of large process structures. In: Proc. CoopIS'07. LNCS 4803 (2007)
5. Reijers, H., Liman, S., Aalst, W.: Product-based workflow design. Management Information Systems **20**(1) (2003) 229–262
6. Aalst, W., Barthelmess, P., Ellis, C., Wainer, J.: Workflow modeling using proclets. In: Proc. CoopIS'00. (2000) 198–209
7. Ryndina, K., Küster, J., Gall, H.: Consistency of business process models and object life cycles. In: Proc. MoDELS'06. (2006) 80–90
8. Redding, G., Dumas, M., Hofstede, A., Iordachescu, A.: Transforming object-oriented models to process-oriented models. In: Proc. BPM'07 Workshops. LNCS 4928 (2007) 132–143
9. Gerede, C., Su, J.: Specification and verification of artifact behaviors in business process models. In: Proc. ICSOC'07. (2007) 181–192
10. Sadiq, S., Orlowska, M., Sadiq, W., Schulz, K.: When workflows will not deliver: The case of contradicting work practice. In: Proc. BIS'05. (2005)
11. Liu, R., Bhattacharya, K., Wu, F.: Modeling business contexture and behavior using business artifacts. In: Proc. CAiSE'07. (2007) 324–39
12. Vanderfeesten, I., Reijers, H., Aalst, W.: Product-based workflow support: Dynamic workflow execution. In: Proc. CAiSE'08. LNCS 5074 (2008) 571–574
13. 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