Improving the Quality and Cost-effectiveness of Process-oriented, Service-driven Applications: Techniques for Enriching Business Process Models

Thomas Bauer
Neu-Ulm University of Applied Sciences, Germany
Stephan Buchwald
T-Systems International, Ulm, Germany
Manfred Reichert
University of Ulm, Germany

ABSTRACT

A key objective of any Service-driven architectural approach is to improve the alignment between business and information technology (IT). Business process management, service composition, and service orchestration, play major roles in achieving this goal. In particular, they allow for the process-aware integration of business actors, business data, and business services. To optimize business-IT alignment and to achieve high business value, the business processes implemented in process-aware information systems (PAISs) must be defined by domain experts, and not by members of the IT department. In current practice, however, the information relevant for process execution is usually not captured at the required level of detail in business process models. In turn, this requires costly interactions between IT departments and domain experts during process implementation. To improve this situation, required execution information should be captured at a sufficient level of detail during business process design (front-loading). As another drawback, existing methods and tools for business process design do not consider available Service-oriented Architecture (SOA) artifacts such as technical service descriptions during process design (look-ahead). Both front-loading and look-ahead are not adequately supported by existing business process modeling tools. In particular, for many process aspects, appropriate techniques for specifying them at a sufficient level of detail during business process design are missing. This chapter presents techniques for enabling front-loading and look-ahead for selected process aspects and investigates how executable process models can be derived from business process models when enriched with additional information.

INTRODUCTION

Business process management (BPM) and process-aware information systems (***PAIS***s) have become integral parts of enterprise computing and are used to support business processes at the operational level (Mutschler, Reichert, & Bumiller, 2008; Reichert & Weber, 2012; Weske, 2007). As opposed to function-

and data-centric information systems, process logic is strictly separated from application code, relying on executable process models that provide the explicit schemes for process execution (Weber, Rinderle, & Reichert, 2007; Weber, Reichert, & Rinderle-Ma, 2008). This enables a ***separation of concerns***, which is a well-established principle in computer science to increase maintainability and to reduce costs of change. A particular challenge for process-aware information systems (PAIS) is *enterprise application integration*, i.e., to link the atomic activities of an executable process model with business functions implemented in heterogeneous, distributed application systems.

In this context, the emergence of Service-oriented architectures (SOA) and well-defined service principles (Erl, 2005) foster ***process-centric*** application integration in the large scale (Weber & Reichert, 2012). A Service-driven approach provides tools for encapsulating heterogeneous application functions as services with standardized interfaces. These services then can be composed and orchestrated in a process-aware manner based on a run-time component called ***process engine***. Altogether, SOA enables enterprise application integration at a high level of abstraction, reducing the need for realizing application-to-application bridges prevalent in current practice (e.g. based on message exchange or remote procedure calls). In particular, any process activity may retrieve data from an application system (e.g. by invoking a corresponding service) and temporarily store it within the process engine. In turn, this data can be consumed, changed, or complemented when executing subsequent process activities (e.g. human tasks).

Another fundamental goal of a Service-driven approach is to improve *business-IT alignment* (Chen, 2008). In particular, a PAIS should meet the needs of the stakeholders involved in the business processes it implements and not reflect only the design decisions made by the IT department. A variety of issues related to business-IT alignment have been investigated in the ENPROSO (*Enhanced Process Management through Service Orientation*) project (Buchwald, Bauer, & Reichert, 2012; Buchwald, 2012). In particular, ENPROSO revealed that all aspects relevant for process automation should be defined at a sufficient level of detail during the design of business process models. Furthermore, existing artifacts should be reused in this context if available. With ***front-loading*** and ***look-ahead***, this chapter presents two fundamental techniques for achieving these goals.

Front-loading enables the capturing of relevant aspects for process automation at a sufficient level of detail during business process modeling. For a ***business process model***, it is not sufficient to only specify the activities, the rough control flow between these activities, and the abstract data objects or roles associated with them. Additionally, the designer of a business process model should capture information relevant for process execution, e.g., about actor assignments (Rinderle-Ma & Reichert, 2007; Rinderle-Ma & Reichert, 2009), user forms (Kolb, Hübner, & Reichert, 2012), and exception handling procedures (Reichert, Dadam, & Bauer, 2003). These and other aspects constitute relevant information for process implementers and therefore should be captured in business process models. They will be discussed in detail in this chapter. In particular, business process models shall be detailed enough to constitute a valuable artifact for process implementers. Note that the latter usually do not have detailed domain-specific knowledge.

Consequently, business process models must not leave room for misinterpretation, such that there is ambiguity in how a business process model is implemented. Only then, it can serve as an appropriate artifact for process implementation. In the current practice, however, many aspects required by process implementers are missing in business process models or only defined in an imprecise manner. This in turn results in significant delays during process implementation due to the need for additional interviews with process stakeholders at a late stage in the project, at which time the domain experts and project consultants involved in the design of the business process might no longer be available. As a direct consequence, in many cases, IT departments themselves decide on how to implement the imprecise or ambiguous specification of a business process model (Mutschler, Reichert, & Bumiller, 2008). This often results in faulty or incomplete process implementations.

Why are business process designers unable to create business process models with a sufficient level of detail fostering their automation in a PAIS? Firstly, domain experts and project consultants are neither aware of the particular aspects relevant for implementing a business process nor the level of detail required in this context, existing Service-driven methodologies do not emphasize this issue adequately. Secondly, business process designers do not have the skill level for defining the IT aspects of a business process. Neither existing process meta-models like ***Event-driven Process Chains*** (EPCs) and Business Process Modeling and Notation (BPMN) nor contemporary business process modeling tools (e.g. ARIS Architect) provide appropriate techniques for defining IT aspects in a way comprehensible for business people and providing the required level of detail. For example, when using ARIS Architect and EPCs, it is not possible to express that a process activity Y must be performed by a person having role "hardware developer" and belonging to the same department as the user who performed the previous process activity X (Figure 1). Usually, business process modeling suites like ARIS only allow assigning a role to a process activity. However, this is not sufficient if more complex actor assignments (Rinderle & Reichert, 2005) need to be implemented.

In order to reduce implementation efforts, available SOA artifacts relevant for process implementation (e.g. technical service descriptions, service implementations, or entities of the organizational model) should be reused and referenced by business process models. We denote the inclusion of respective references in business process models as *look-ahead.* Existing Service-driven methodologies, however, do not provide techniques enabling such a look-ahead. Indeed, integrating SOA artifacts with business process models constitutes a non-trivial task. This is not surprising considering the fact that business process designers do not have the skills required to handle the technical specifications of IT artifacts. For example, a service implementing a particular activity of a business process model is usually described in a technical style. However, specification languages like WSDL are not comprehensible to business people having no or only little IT background. Also note that textual annotations of service descriptions, as provided by many IT departments, will be hardly comprehensible due to the large discrepancies existing between technical specifications and common business languages (e.g. technical artifacts like XSD data types are used to describe the input / output parameters of a service).

Front-loading and look-ahead can be applied to different process aspects (Reichert & Weber, 2012), including *control flow* (e.g. flexibility by design or inclusion of exception handlers), *data flow* (e.g. data types of process data elements), and *organizational entities* (e.g. actor assignments). However, techniques enabling business process designers to specify technical aspects or reference IT artifacts during business process modeling are still missing. This chapter analyzes the requirements for look-ahead and frontloading, and presents selected techniques in detail. In particular, the techniques suggested can be included in a corporate SOA methodology. For example, a governance committee responsible for the quality of business process models may only release models enabling front-loading and look-ahead. (Buchwald, 2012) presents a corporate SOA methodology for PAIS development, which covers front-loading and look-ahead as well. In early phases during business process design, modelers must capture front-loading data and refer to existing SOA artifacts to realize look-ahead. In a subsequent phase, respective information is then used for the technical specification and implementation of executable process models.

The remainder of this chapter is structured as follows: We first provide background information and introduce a scenario used throughout the chapter to illustrate basic issues and concepts. We then analyze for which process perspectives front-loading and look-ahead are useful. In this context, we elaborate general requirements and present front-loading and look-ahead techniques for selected process aspects. In particular, these techniques can be used by business process designers having only few IT skills. Front-loading will be illustrated for actor assignments, where we will present techniques for precisely specifying complex actor assignments during business process design. In turn, look-ahead techniques will be illustrated along with the integration of service implementations in business process models. In this context, the challenge is to enable business process designers to find required services based on available

documentation described at the business level. For both scenarios, we discuss how respective information can be utilized during process implementation. The chapter concludes with a summary and outlook.

BACKGROUND

The framework developed in the ENPROSO project comprises several methods that improve business-IT alignment when implementing business processes in a Service-driven environment. One of these methods bridges the gap between business process models on one hand and business process implementations (i.e. process orchestrations) as well as related technical specifications on the other (Buchwald, Bauer, & Reichert, 2012). For this purpose, different models are used.

One of these models – denoted Business-IT-Mapping Model (***BIMM***) – captures the dependencies between the steps of a business process model (e.g. specified in terms of an EPC) and the steps of the corresponding system process (i.e. technical implementation). As illustrated in Figure 1, a single step of a business process model may be realized in terms of multiple steps at the IT system level (i.e. mapping type "Split" is used). As example, consider an activity that allows a user to enter data via a form and then automatically store this data in an IT system. Such an activity will be implemented as a human task realizing the user interaction, followed by an automatically invoked service that writes the data into the respective IT system.

Note that a system process may contain additional steps not present in the corresponding business process model, such as using a mapping type "Insert", or performing a technical step such as writing data to a system log. In many cases, however, a business process step is just mapped to exactly one step of the system process. In this context, mapping type "Rename" must be applied if different labels shall be used for the respective step at the business and system process level. Mapping type "Merge" aggregates multiple steps of the business process to a single step of the system process. This mapping is useful in cases where the same person must perform a sequence of separate business activities to be realized by one and the same human task (in the implemented system process) to minimize required user interactions (Kolb, Hübner, & Reichert, 2012).

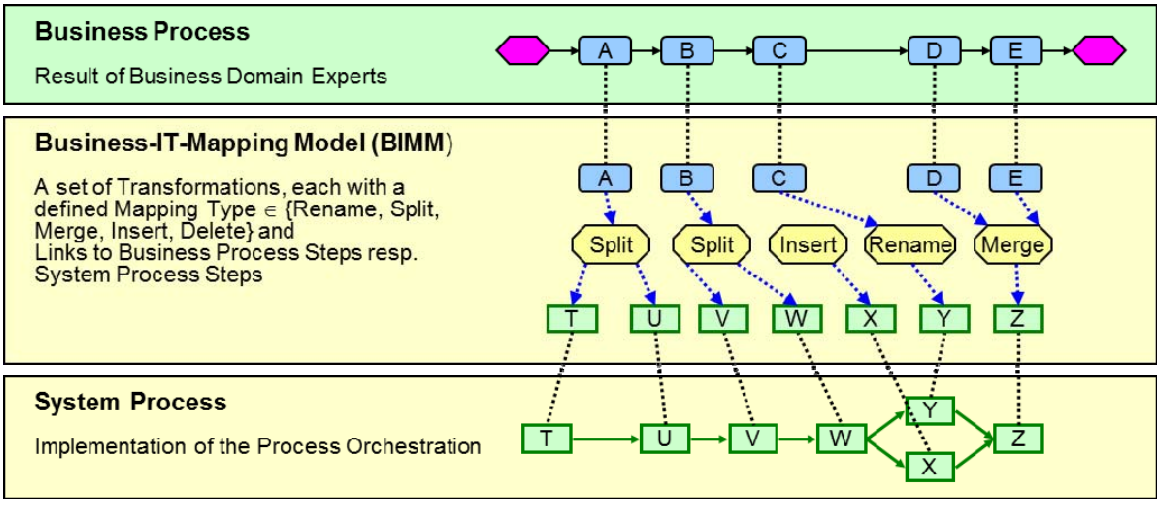

Figure 1. *Business-IT-Mapping Model capturing dependencies between process steps at different levels.*

In general, a BIMM represents all dependencies between the steps of a business process model and that of the corresponding system process model (Figure 1). Based on respective mappings, for each business

process step (e.g. A) the corresponding steps at the system level can be identified (e.g. T and U). Especially, when business requirements related to a business process step evolve, it will be easy to figure out which system level steps might have to be adapted. In summary, when using a BIMM, changes in business requirements may be quickly transferred to the operational level with reduced effort and cost compared to existing approaches.

Since a BIMM allows mapping the information defined at the business process level to elements of the system process, it can be used as the basis for realizing *front-loading* and *look-ahead* as well. If business level information is changed after the first version of the business process model was released and process implementation has already started, the BIMM enables quick and easy identification of those steps of the system process affected by the change. For example, assume that the *actor assignment* of business process step A (Figure 1) is changed. Based on the given BIMM, it can be easily figured out that A is implemented by steps T and U in the system process. Assume further that only T represents a human task, whereas U corresponds to an automated service storing data. In this scenario, only the actor assignment of step T must be changed in the system process.

Related Work

Both front-loading and look-ahead are methods known in other domains as well. In the automotive industry, for example, *product engineering processes* apply *front-loading* to achieve a higher maturity level of the product in the early development phases. For this purpose, selected development activities are performed earlier than required. Based on existing, but still evolving CAD (Computer Aided Design) models of vehicle parts, it is checked whether the production of the vehicle might cause problems in future lifecycle phases (e.g. when producing the vehicles in the factory). *Look-ahead* is realized by investigating which vehicle parts (e.g. a specific electric motor) have been already used by other vehicle components or in other vehicle engineering projects and, therefore, might be reused in the current project.

With contemporary business process modeling tools, front-loading and look-ahead can be partially realized for selected process aspects (Enderle, 2009). For example, *flexibility flags* marking process regions with high flexibility demand (Figure 2) can be realized by assigning *annotation objects* to the respective process steps. As will be shown, however, existing business process modeling tools do not provide an appropriate basis for business process implementation due to their restricted functionality as well as the incomplete and ambiguous specifications provided by them (Figure 1).

Generally, a Service-driven methodology should enforce both *front-loading* and *look-ahead*. In the following text, we will analyze Service-driven methodologies with respect to specific techniques.

SOMA (Service-Oriented Modeling and Architecture) enforces front-loading for some of the process aspects discussed in this chapter (Arsanjani et al., 2008). For example, flexibility flags can be realized during the identification phase. In turn, exception handlers and data objects may be defined during the specification phase. However, SOMA does not allow for querying existing service implementations to enable their reuse in business process models (look-ahead). Finally, front-loading in terms of early specifications of complex actor assignments or forms for human tasks is not supported.

Quasar Enterprise (Engels et al., 2008a; Engels et al., 2008b) allows adding flexibility flags to simple business process models as well. Like in SOMA, business rules can be used for this purpose. However, creating such rules requires special skills that business process designers usually do not have. Querying services based on business criteria and hence reusing services based on look-ahead are not supported.

M3 for SOA (Deeg, 2007) constitutes a modeling method that uses different model types in the different phases of the process life cycle (Weber et al., 2009). However, it is not possible to define organizational aspects at a sufficient level of detail during business process design (initiation phase). Furthermore, data objects and business rules may be specified within a business process model although this is not directly

supported in existing business process modeling tools. Look-ahead for existing data objects and services is not supported at all.

AVE (ARIS Value Engineering) (IDS Scheer, 2005) provides a methodology for BPM projects that is strongly connected with the ARIS toolset and EPC. Hence, for many process aspects, a sufficient (i.e. complete and unambiguous) support of front-loading or look-ahead is not possible. For example, complex actor assignments cannot be specified unambiguously when using AVE (Figure 1). Further, querying services in the service repository based on business criteria is only rudimentarily supported.

SOAM (SOA Method) allows for querying services and reusing them (Offermann, 2008; Offermann & Bub, 2009). However, other process aspects discussed in this chapter (e.g. front-loading through adding flexibility flags or precise actor assignment to business process models) is not supported. Other SOA approaches like OrVia (Stein et al., 2008), Project-Oriented SOA (Shuster, 2008), and SUPER (Weber et al., 2007) provide methodologies for developing process-oriented applications in a SOA (process applications for short). However, they only rudimentarily support the front-loading and look-ahead techniques presented in this chapter.

In summary, neither existing business process modeling tools nor SOA methods meet the requirements raised by front-loading and look-ahead in a sufficient manner. Some approaches enable front-loading of individual process aspects (e.g. actor assignments or flexibility flags). During business process design, however, these aspects can only be defined at a rather abstract level, which hinders their direct realization during process implementation. Furthermore, SOA literature does not report on methods enabling front-loading or look-ahead. In particular, appropriate modeling techniques are still missing.

Application Scenario

Figure depicts a business process dealing with product change management in the automotive domain. This simplified process will be used as an illustrating example throughout this chapter. Its purpose is to allow responsible users to request a change of a product (e.g. a vehicle) in its development or production phase. The requested change must be described and analyzed, its costs must be estimated, the request must be approved or rejected, and finally the change must be realized (in case of approval).

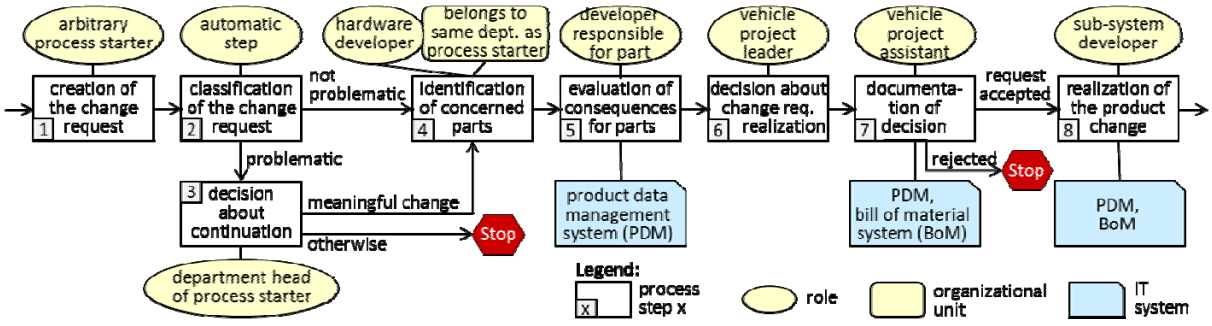

Figure 2. *Product change management process.*

In Figure , the first process step (i.e. Step 1) deals with the creation of a new change request. This step may be performed by an actor who is allowed to start instances of the product change management process. Furthermore, the respective actor must provide information about the product change desired in Step 1. For this purpose, the actor edits a form and enters the required data, such as the reason for the change request, a technical description of the change, an estimation of resulting costs and earnings, the deadline for realizing the change, and concerned objects (e.g. prototypes, production plants). Steps 2 and

3 are then performed to prevent unnecessary or costly change requests. More precisely, Step 2 constitutes an automated process step that classifies the change request according to the data provided by Step 1.

A change request will be considered as problematic, for example, if its estimated costs are too high, its deadline cannot be met, or it refers to a vehicle project that has already entered the production phase. For problematic requests, Step 3 is additionally performed, i.e., the manager (e.g. department head) of the change requestor then decides whether the processing of the change request shall be continued (e.g. for safety reasons). If a problematic change request is not considered as meaningful, it will be stopped in this early phase, otherwise it will be continued. Change requests classified as "non-problematic" in Step 2 will be continued anyway. Note that Step 2 reduces overall efforts for the department head, since he/she does not need to handle unproblematic requests.

The technical evaluation of the change request starts with Step 4. During this process step, a specialist identifies all product parts concerned by the requested change. This specialist must have the role of "hardware developer" and belong to the same department as the change requestor (i.e. the process starter). Step 5 is then executed multiple times in parallel according to the number of parts affected by the change. For each part identified in Step 4, required technical changes, changes in product weight, estimated costs, etc. are evaluated. Thereby, each evaluation is performed by the developer responsible for the respective part.

As a prerequisite for Step 5, detailed information about product parts needs to be provided, e.g., by invoking a respective information service offered by the product data management system (PDM). After completing all part evaluations, in Step 6 the project leader decides whether or not the change shall be approved. In Step 7, a project assistant documents this decision, which is then automatically logged in several IT systems. To prevent undesired changes, change approval constitutes a prerequisite for changing part data (e.g. geometry data) in these systems. Following this, rejected change requests are stopped, whereas approved ones are realized by executing Step 8. The latter is performed multiple times by developers responsible for the concerned sub-system for which parts or entire components need to be changed. Finally, resulting data is stored in the PDM system and the BoM (Bill of Material) system.

REQUIREMENTS FOR FRONT-LOADING AND LOOK-AHEAD

This section analyzes the process aspects for which front-loading techniques are useful. In this context, we refer to the problems that will arise if respective aspects are not already specified during business process design. Furthermore, for each process aspect, we show an appropriate front-loading technique. Finally, general requirements for front-loading are summarized. Applying the same schema, issues related to look-ahead are presented. Selected approaches enabling front-loading and look-ahead are presented in the subsequent section.

Process Aspects for which Front-Loading is Useful

To better understand the benefits provided by front-loading, for the different aspects of a business process, we analyze the information that would typically be captured during business process design.

Organizational Aspect

In a business process model, an actor assignment defines the persons who may perform a particular process step (Rinderle & Reichert, 2005). In a later development phase, these actor assignments are then implemented at a technical level. During run-time, a process engine ensures that instances of this process step are only added to the work lists of authorized users. That means, only the persons that qualify for the process step (i.e. belonging to the actor set specified by the respective actor assignment) are allowed to perform this step. Therefore, the precise and complete specification of actor assignments at the business

level is crucial for correctly and efficiently implementing them in a PAIS. However, the concrete implementation method depends on the process engine chosen. While certain process engines require program code in this context, others provide more sophisticated support for specifying actor assignments in terms of pre-defined templates whose parameters may be linked to entities (e.g. roles) of the organizational model of the company (e.g. the corporate directory) as well as the process instance data (e.g. to refer to the actual performer of a previous process step). At run-time, the process engine then uses the organizational model of the enterprise to determine the potential actors of a process step. An example of a process engine enabling this approach is AristaFlow BPM Suite (Dadam & Reichert, 2009; Reichert et al., 2009).

To better understand the challenge of defining actor assignments during the design of a business process model, we discuss how respective expressions may be specified when using a contemporary business process modeling tool. More precisely, for the business process model depicted in Figure , Figure 1a shows how the actor assignment of Step 4 can be expressed when using the ARIS modeling tool; remember that this step shall be performed by a person owning the role of "hardware developer" and belonging to the same department as the performer of the first process step (i.e. the change requestor).
Usually, a process step within an ARIS model only refers to the role an actor performing this step shall have. In some cases, in addition, "department objects" with unclear semantics are used (Figure 1a).

However, the information provided in this context is not sufficient to implement the process step in a way required to execute it by a process engine. As indicated by the dotted lines in Figure 1b, the exact department that the hardware developer belongs to is determined during run-time by taking the execution history of the process into account. With business process modeling tools like ARIS, however, it is not possible to express such inter-dependencies relevant for process execution. In fact, ARIS does not support any object type representing the actual performer of a particular process step. Furthermore, there is no relationship type allowing the process model designer to specify that the department assigned to Step 4 shall be the one a particular actor (e.g. the performer of Step 1) belongs to.[1]

Note that there exist many similar scenarios not adequately covered by tools like ARIS due to the insufficient expressiveness of their business process modeling language. For example, in ARIS it cannot be expressed that a change request shall be approved by the head of that department to which the change requestor belongs, i.e., it is not sufficient to only assign a role "department head" to the respective process step (i.e. Step 3 in Figure), but to express this dependency as well. Similarly, a bank clerk applying for a credit must not approve this credit request or that a particular document must be checked by a person different from the one who created it (i.e. separation of concerns). In the latter example, again it might be necessary that the second specialist belongs to the same department as the first one (Figure 1b).

Regarding Step 4 in the model from Figure 1, an actor assignment like "role = 'hardware developer' AND department = department(performer(Step 1))" is required. In particular, incomplete information like "role = 'hardware developer'" (Figure 1a) prohibits the correct implementation of the desired behavior since the actor set corresponding to this expression would be too large (i.e. the set would include all hardware developers from arbitrary departments). In the subsequent section, several modeling techniques

[1] It is possible to add the name of the referenced process step 1 to the department object. In our example it may have name "same department as performer of step creation of the change request". Referencing the name as text (instead of an explicit edge), however, is not a proper modeling technique and may result in errors. In particular, such a text cannot be interpreted correctly if the name of the process step is changed in future. Additionally, this name is not unique if the same activity is added multiple times to the same business process model.

will be presented, which allow defining such complex actor assignments during business process design without requiring comprehensive IT skills.

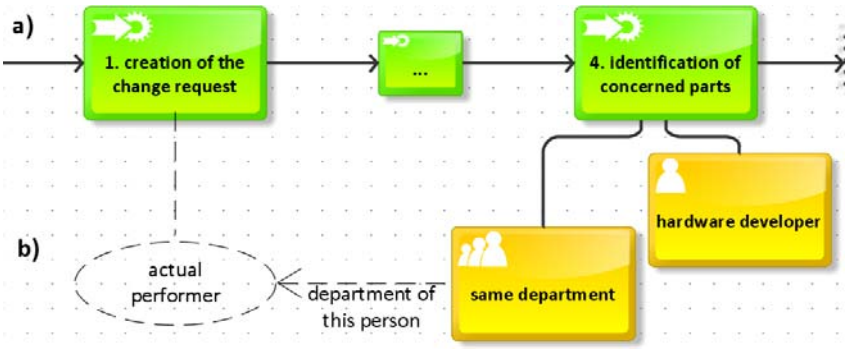

Figure 1. *An actor assignment defined a) with ARIS and b) including the additionally required dependencies.*

Substitution rules constitute another element of the organizational perspective (Bauer, 2009). Basically, a substitution rule defines the actors who shall take over a particular process step in case its regular performers are absent (e.g. due to a business trip, sick leave, or holidays). Most process engines support substitution rules and hence prevent process steps from being solely assigned to the worklists of absent persons. Note that the latter might cause significant delays as well as high costs (e.g. due to deadlines missed) during business process execution. As a prerequisite for implementing substitution rules, however, for all processes and process steps, respectively[2], domain experts must specify which persons shall act as substitute. For example, consider Step 6 in Figure . From the information provided, it does not become clear for implementers who shall act as substitute of the vehicle project leader. Perhaps there exists a vice project leader or decisions are made by a project committee. Alternatively, the supervisor of the project leader might have this responsibility. The techniques we will present for defining actor assignments during business process design can also be used to realize front-loading of substitution rules.

An escalation rule defines notifications sent to specific actors or managers if a process step (representing a human task) is not started or completed within a pre-specified time period. In certain cases, it might be even required to automatically forward a delayed human task to another actor. For example, consider Step 5 (evaluation of consequences for parts) from Figure , for which multiple instances are concurrently processed by different developers. Manually monitoring these instances and their temporal constraints (Lanz, Weber & Reichert, 2010; Lanz, Weber, & Reichert, 2012) by a process administrator would require a huge effort due to the potentially large number of human tasks (even in the context of a single change request). Instead, a process engine should allow for their automatic monitoring, and automatically trigger escalations in case of delays. Again, this necessitates the provision of respective information during business process design, i.e., domain experts should specify the escalations required in a specific context (e.g. violations of temporal constraints). Generally, it does not constitute a valid option to allow process implementers realizing escalation rules in an arbitrary way. This becomes obvious when

[2] In general, substitution rules should not be defined globally for a person or a business process. Instead, a substitution rule is necessary for each single process step (cf. Bauer 2009). For example, consider a team leader who is substituted by an experienced team member at a step in the change request process (e.g. step 5 in Figure 2: evaluation of consequences for parts). Regarding executive functions (e.g. approval to increase a budget), however, it is substituted by another team leader or even by his supervisor.

considering the fact that in large companies, technical implementations are often outsourced to external contractors. However, the latter may have only low economic interest to invest any effort in clarifying ambiguities with domain experts.

Altogether, front-loading is essential for all issues related to the organizational aspect.

Data Objects

The data objects of an executable process model have associated data types to precisely define the interface of the process model as well as its internal data flow (Reichert & Weber, 2012; Künzle, Weber, & Reichert, 2011). Usually, data is provided by the environment when starting a process instance. In turn, when completing a process instance, it may return output data back to its environment. Furthermore, when a process instance invokes a service during run-time (i.e. to execute a process step), input data required by this service are consumed from process data objects, whereas output data produced upon service completion are stored in corresponding data objects (Reichert & Dadam, 1998). For example, consider Step 5 of the process model from Figure , which invokes a service to read part data from a product data management (PDM) system. Thereby, the process provides the part number as input to enable the invoked service to identify the part concerned by the change and to return all part data required by the process.
A human task is often implemented as an electronic form (Kolb, Hübner, & Reichert, 2012; Künzle & Reichert, 2011). Input data types then represent the information presented to the actor processing the form, whereas output data types correspond to the data that may have to be set for updatable form elements. Usually, data types are only roughly specified in process models at the business level, e.g., only the name and most relevant attributes may have been captured. This information, however, is not sufficient for implementing the interfaces of executable process models and callable services (including user forms) respectively.

Regarding data objects, front-loading may contribute to avoiding unnecessary analyses during process implementation. For this purpose, for each data object, all attributes (including their data types) relevant from a business perspective must be specified. Regarding our running example, for instance, this means that a domain expert must decide whether the part number assigned to Step 5 (and the service it invokes) shall have type integer or string, e.g., to store part numbers like “4711” or “C-284285-2012”. Note that only domain experts will have the knowledge required in this context; e.g., the first character of a part number encoding the vehicle project (e.g. C-Class) and the last 4 digits encoding the development year of the part. Hence, data type string (with a length of 13) must be assigned to the respective data object in the business process model. Finally, for each process step, data objects read or written must be described at a similar level of granularity to avoid further analyses or ambiguities in the process implementation phase.

Services

When a process step is executed, it invokes a service, hands over data objects to it, and receives service results that may change the existing process data. To enable front-loading, the services required for implementing the process must be described in detail within the business process model, i.e., it is not sufficient to only define service names as placeholders. For example, assume that Step 7 in Figure requires a not yet implemented service documenting the decision about a change request in the BoM system. In this case, the service name itself (e.g. BoM_StoreChangeRequestDecision) will not be sufficient to implement the service. In addition, the functions to be provided by the service need to be specified, e.g., (i) storing the number of the change request in the BoM system, (ii) assigning the parts affected by the change to this number, and (iii) allowing for future modifications of this part if the correct change request number is referenced.

In general, for each service to be invoked by a process, its functionality, data types of its input and output parameters, and desired quality of service (QoS) properties like response time (Bodenstaff, Wombacher, & Reichert, 2008) need to be pre-specified. A process implementer may then use this information either to select an existing service or to develop a new one. In addition, the business process designer should specify which IT system shall offer this service. Regarding Step 5 from Figure , it makes a bit of a difference whether data is retrieved from the PDM or the BoM system.

Note that these two systems store different attributes for product parts. Furthermore, in a BoM system, new part versions only become available with delay (i.e. when reaching a higher release level). There exist process modeling tools that already allow defining services from a business perspective, to specify desired service functions and properties. To foster front-loading, respective tool support must be provided to domain experts during business process design. Finally, this must be ensured by an appropriate SOA methodology (including governance processes).

Flexibility

When implementing a business process, those points or regions need to be known for which a high degree of flexibility is required during process execution (Sadiq, Sadiq, & Orlowska, 2005; Weber, Sadiq, & Reichert, 2009; Reichert, Rinderle-Ma, & Dadam, 2009). Since these flexibility points differ from process to process, they need to be identified and specified by a domain expert together with the required kind of flexibility. For example, there might be process steps whose associated service implementation frequently needs to be exchanged due to evolving business requirements. Regarding Step 2 in Figure , for instance, there exist frequently changing criteria used for deciding on whether or not a change request is problematic. Since Step 2 constitutes an automated step, the decision algorithm is implemented by a corresponding service. In particular, this service implementation should be exchanged whenever requested by the business domain, if the decision algorithm or its underlying criteria change. Hence, a flexible implementation of the service call is required that allows exchanging the service implementation if required.

In practical settings, domain experts usually know the services whose implementation frequently changes. Hence, front-loading can be realized by flagging the process steps that use these services and annotating them with respective descriptions. Based on this information, it becomes clear that the service call needs to be implemented in a decoupled manner, such as, by using an enterprise service bus (ESB) or a message broker. Other process artifacts that may require a similar degree of flexibility (i.e. dynamic exchangeability) include branching conditions, business rules, actor assignments, and timeout limits. In particular, it must be possible to add flexibility flags and descriptions to these artifacts during the design of a business process model, if required.

As an example consider Figure 2, which shows a process step with an associated flexibility flag and description. More precisely, the latter indicates that the service implementation of this process step frequently changes. When using a contemporary process modeling tool like ARIS, respective flexibility flags and descriptions can be realized by creating a corresponding object type for them (or by deriving such an object type from an existing one if the tool does not allow creating new object types). A particular attribute of this object type should then encode the required kind of flexibility. Furthermore, it should be possible to describe the expected changes by using a documentation or comment attribute as usually offered by any process modeling tool for all object types. The flexibility requested in Figure 2, for example, can be implemented by specifying a corresponding business rule using a business rule engine and connecting it with the process implementation. Consequently, changes can be directly mapped to rule modifications without requiring any adaptation of the process implementation itself.

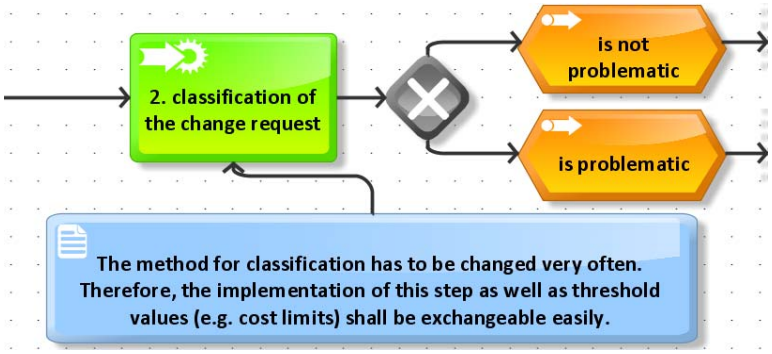

Figure 2. *ARIS process step with a flexibility flag and description.*

Exceptions

During the execution of a business process, it might become necessary to handle exceptional situations (Reichert & Weber, 2012). Examples of exceptions include failed service calls (e.g. the PDM system called by Step 5 might be not running), insufficient data quality (e.g. the data entered by a user in the context of a human task, e.g., Step1, might be incomplete or contradictory), and empty actor sets (e.g. no actor can be assigned to Step 4 after the only hardware developer of the respective department quit his job). Obviously, some of these exceptions can be anticipated and hence be automatically detected (e.g. failed service calls). Consequently, the IT department can detect them without need for domain-specific knowledge and no front-loading is required for exception detection. By contrast, insufficient data quality can be only detected if additional information is made available by a domain expert. For example, to detect that attribute values “vehicle project = Actros” and “part number = C-284285-2012” are contradictory, one must know that Actros is a commercial vehicle, while part numbers of kind “C-number-year” belong to the passenger car project C-Class.

While many exceptions can be detected without need for specific domain knowledge, exception handling itself usually requires domain knowledge and hence should be incorporated into business process models using front-loading. For example, a failed service call might be repeated infinitely in worst case. However, if a service call fails repeatedly, a domain expert should decide whether it should be repeated a certain number of times, an alternative service (e.g. causing higher costs) should be called, or a human task should be performed instead, to create service results manually. Thereby, the domain expert has to balance between the execution delays and costs incurred by the options available for exception handling. Similarly, if contradictory attribute values occur, it must be defined who shall correct the inconsistencies. Finally, if the set of potential actors for a human task is empty, a process manager (i.e. a responsible actor) is required who manually re-assigns the task to an actor being able to perform it.

Forms

Usually, a user performing a human task fills in a respective form. As mentioned, the data fields displayed to the user or edited by the user are defined by the data types of the input and output data objects of the process step (Kolb, Hübner, & Reichert, 2012). For example, assume that the input data object of Step 3 (see the process model from Figure) corresponds to the change request produced in Step 1. Then the attributes of this data object will be displayed in respective form fields to the responsible department head and provide the basis for their decision.

Generally, a data object and data type respectively is used in the context of multiple process steps, i.e., its content is not optimized for a specific process step. Consequently, a data type often comprises more

attributes than required or useful for performing a particular process step. In such a case, not all attributes shall be displayed to the performer of the process step (Künzle & Reichert, 2009; Künzle & Reichert, 2011). Regarding Step 3, for example, the department head shall only see selected attributes of the change request relevant for the decision, but not all technical details of the change as captured in the change request data object as well.

Regarding the user forms assigned to process steps, front-loading means that a domain expert must define which attributes shall be displayed to the user and which not. The same applies to form fields and data object attributes respectively that must be provided by the user. Generally, it is the task of the process designer and domain experts to decide which data object attributes shall be read or written in the context of a particular process step, i.e., respective design decisions should not be made by the implementer of the user form. As opposed to technical process specifications (Reichert & Dadam, 1997; Reichert & Dadam, 1998), in most existing business process modeling tools, a data type does not contain information about mandatory and optional attributes. As a consequence, for an output data object of a process step, it is not always clear whether a particular attribute shall be realized as mandatory input field in the form.

For example, in Step 3 the department head may provide the decision as a Boolean value indicating whether to stop or continue the processing of the change request. Additionally, the department head may comment on the change request. Using front-loading, for instance, the domain expert may specify that the Boolean value representing the user decision is mandatory and hence shall be implemented as mandatory form field, whereas the comment shall be realized as optional field and certain attributes of the change request object shall be omitted. Regarding process steps, in general, for each attribute of its output data objects, a corresponding status $\in$ {mandatory, optional, omit} shall be defined in the business process model. In turn, for each attribute of an input data object a status $\in$ {show, omit} is required.
Besides the contents of a form, its design is crucial as well (Kolb, Hübner, & Reichert, 2012). Hence, front-loading should allow domain experts to provide information about the desired appearance of the form as well. Form implementers shall be supported by attaching drawings to process steps that define the positions of the data fields within the respective form as well as the labels to be displayed for the different form fields. Additionally, such a drawing might define the GUI control elements (e.g. text field, combo box, list, radio button, etc.) that shall be used for interacting with specific form fields and data objects respectively.

Other process aspects

Other process aspects, for which front-loading is useful, include transactional properties of a business process model and measurement points enabling process performance management. We refer to (Enderle, 2009; Buchwald, 2012) for further details.

General Requirements for Front-Loading

Front-loading is a useful concept for a variety of process aspects and hence different techniques enabling it exist. Besides its particularities, which meet the requirements of a specific process aspect best, any front-loading technique should satisfy a number of general requirements:

- It must allow modeling the information required for implementing a specific process aspect from a business perspective; respective information shall be provided by a domain expert, to foster correct process executability, and increase process model quality.
- It must be comprehensible and easy to use for business process designers having no or only limited IT skills. In addition, the artifacts resulting from its use must be easy to read, i.e., they should be understandable to domain experts not having process management skills. Note that this is crucial for verifying the semantic correctness of the business process models.

- The specifications (i.e. artifacts) resulting from the application of any front-loading technique must be complete, i.e., their information content must be sufficient to implement the respective process aspect later on without need for costly interactions with domain experts.
- For each front-loading technique, a procedure shall be provided that allows transforming the respective business specification into a correct IT implementation. In principle, such a transformation shall be as efficient as possible. However, automatic, semi-automatic, and manual transformations have to be considered depending on the given application environment.

Process Aspects for which Look-Ahead is Useful

We describe the process aspects for which the existence of appropriate IT artifacts shall be checked during business process design in order to enable the reuse of these artifacts and hence to reduce process implementation efforts.

Organizational Aspect

As described in the context of front-loading, a business process model shall capture actor assignments, substitution rules, and escalations. Usually, respective information is defined in terms of expressions referring to organizational objects (OrgObjects for short). Examples of the latter include roles, departments, and competences. In a later development phase, i.e., during process implementation, these expressions must be transformed in a way that the resulting specifications are machine-readable, referring to real objects from the organizational model of the enterprise. During process execution, the organizational model is then used to determine potential actors of the respective process step. For this purpose, this model maintains all OrgObjects and their relations, as well as the assignment of concrete actors to them.

For example, to determine the potential actors of Step 4 in Figure , information about the employees, currently owning role “hardware developer” and belonging to the respective department, is utilized.
In general, the transformation of a business process model into an executable process implementation must not leave room for ambiguities. This necessitates that the OrgObjects referenced by a business process model are unambiguous, e.g., the label “hardware developer” used in the context of Step 4 should refer to a unique role. Ideally, a business process designer uses exactly the same names for OrgObjects as known in the organizational model of the enterprise. Concerning the organizational aspect, this would ease the transformation from a business process model to an executable process implementation significantly. In turn, when using varying names for existing OrgObjects at the different levels, during process implementation, it might not be clear which OrgObject shall be actually referenced.

As a particular challenge, OrgObjects may correspond to IT artifacts not known and not self-explanatory to business process designers. Hence, additional information is required to enable these experts to refer to entities (i.e. OrgObjects) from the organizational model of the enterprise and hence to reuse them if favorable. For this purpose, the structure of the organizational model (e.g. object types like actor, role, department, and competence, as well as the relationship types between them) needs to be published in a way easily comprehensible to business process designers. Furthermore, the concrete instances of the different OrgObjects[3] must be published, e.g., the concrete roles or department names that may be used in a business process model (e.g. “hardware developer”, “component architect”). Finally, appropriate support for querying OrgObject instances in the organizational model should be provided. Generally, such look-ahead increases the unambiguousness of a business process model. Further, it improves reusability of organizational objects, avoiding their redundant specification and hence reducing maintenance efforts.

[3] At business process design, normally, no concrete persons shall be referenced. Therefore, it is not necessary to publish information belonging to individual persons (e.g. their assignment to roles).

When creating a business process model, even OrgObjects that do not exactly match the given requirements might be used. For example, consider Step 8 in Figure and assume that OrgObject “sub-system developer” currently does not exist in the organizational model. A domain expert, however, might have the knowledge that all sub-system developers either possess role “component architect” or role “hardware developer”. If both roles already exist in the organizational model, role = component architect OR role = hardware developer should be chosen as actor assignment to enable the reuse of existing IT artifacts.

Data Objects

When designing a business process model, data objects are used for specifying the data flow between process steps, e.g., the change request object produced by Step 1 (Figure) is consumed (i.e. read) by Step 2. Furthermore, data objects provide the basis for transmitting data between a process instance and invoked services (e.g. Step 5) or human tasks (e.g. Step 6). To enable look-ahead for the data perspective, existing data object types should be reused during the design of a business process model as well. Usually, for respective data object types, there already exist detailed specifications (e.g. UML class diagrams) or even implementations (e.g. Java class implementations). To enable the reuse of existing data object types through look-ahead, these should be easily accessible to business process designers and respective tools. As an advantage, efforts for designing, specifying, and implementing data objects are significantly reduced.

Even more important, business process models then refer to the same data types as existing services and human tasks. Due to this harmonization, the overall efforts for implementing the transformation of a data structure defined at the business level into another one used at the technical level can be avoided, such that, both the change management process and the service interfaces provided by the PDM system may refer to the same data type in the context of part data objects. Regarding our example from Figure , Step 5 calls a service of the PDM system to retrieve required part data. Due to the harmonization that can be achieved through look-ahead, there is no need for transforming the received part data, neither during the service call nor during business process execution.

Services

As discussed, a business process model may call a service to trigger an action (e.g. data delivery) in a foreign IT system. Basic to this are the service interfaces that allow other applications to call specific service operations. Usually, service descriptions and service operations are published in a SOA repository. At process execution time, the SOA repository may then be used by an Enterprise Service Bus (ESB) to figure out the service endpoint before calling the service. Such a decoupling increases flexibility since it allows moving a service to another server without the necessity to modify all applications consuming this service. Furthermore, even changes of service signatures become possible, since an ESB may transform input or output data during the service call as well. All information required for endpoint lookup or transformation may be stored in a technical SOA repository like IBM WSRR (IBM, 2007). SOA repositories often manage large data volumes since they have to store information about service versions, business objects, usage contracts, and many other object types as well as their dependencies. A comprehensive meta-model of a SOA repository is presented in (Buchwald, 2012). Furthermore, SOA repositories are relevant in the context of semantic web services as well.

Regarding our running example, a SOA repository might be queried to search for already existing services that allow realizing Steps 5, 7 and 8. Unfortunately, services and their operations are often published only at a very technical level, e.g., using WSDL descriptions. However, such descriptions are not comprehensible to business process designers having no or only limited IT skills. Similar problems hold for textual service descriptions as offered by many IT departments as alternative to WSDL

specifications. Usually, the language used in this context significantly differs from the one used by domain experts, especially when technical artifacts (e.g. XSD data type specifications for service parameters) are used.

To enable look-ahead, a service should be described in terms of a language (or graphical notation) easily understandable to business process designers. Furthermore, it must be easy to search for needed services and to access and understand related descriptions. Based on this, business process designers can find appropriate services and refer to them in corresponding process steps. Note that existing technical service specifications (e.g. service operations and the data types of their input/output parameters) then should be linked with these business level specifications to avoid unnecessary analysis and inquiries during process implementation.

For some process steps, no service might exist that precisely offers the required functionality. In such a situation, it might still be useful during business process modeling to refer to a service having similar functionality. For example, assume that for Step 5 in Figure , there exists no service delivering the part data based on the “part name” as input. Assume further that the PDM system already offers a service delivering respective data based on the “part number”, which is not known within the change management process when executing Step 5. Before calling the respective service, therefore, the “part number” must be figured out based on the given “part name”. This can be accomplished by introducing an automatically executed service or an interactive process step (human task) in the executable process model. Due to these changes in process behavior, this decision should be made by a domain expert during business process design.

To enable service look-ahead, existing services should be published and guidelines must be defined that enforce their reuse if favorable from an economic point of view. Usually, the reuse of existing services makes sense, and it does not only reduce efforts and costs for service implementation, but also eases service maintenance significantly. As a disadvantage, adjustments to the defined process logic might become necessary (see the example above). In certain cases, the costs resulting from such a restructuring are negligible, for e.g., if the sequence of already existing process steps is changed (e.g. determining the “part number” earlier as intended) or process steps (i.e. service calls) are added that can be completely automated (i.e. no additional human interactions become necessary). Since domain experts are responsible for the design of business process models, they might also decide that an existing service shall not be re-used and a new one be requested instead, e.g., if costly human tasks become necessary or process quality suffers.

Business Rules

Business rules are used for implementing complex branching decisions within business processes. Step 2 from Figures 2 and 4, for example, automatically classifies a change request either as “problematic” or “not problematic”. For problematic change requests, in addition, Step 3 must be executed in order to decide whether the processing of the change request shall be continued. Since the logic of the rule for deciding whether a change request is problematic might be rather complex and frequently change, it can be implemented using a business rule engine. The latter maintains business rules in a central repository and provides comprehensive support for rule execution. Even more important, by extracting complex rules from business process logic and implementing these rules separately in a business rule engine, rule changes can be accomplished without need for changing or re-deploying business processes and their implementation. Furthermore, sophisticated tools for defining or editing business rules are provided. Using these tools, domain experts without deep IT skills will be able to perform changes of a business rule (e.g. changes of the thresholds defined in the context of a business rule).

Look-ahead fosters the reuse of business rules. In particular, this helps avoiding unnecessary costs due to redundant implementation of already existing business rules. Note that such redundancies increase rule maintenance costs significantly and result in inconsistencies as well as unclear process behavior when business processes use different (and potentially degenerated) implementations of the same rule. For example, assume that there exists a business rule enabling customer classification (i.e. it characterizes a customer as standard, premium, or problem customer). There should be exactly one implementation of this classification rule that may then be reused in a variety of business processes. Only then, ambiguities can be avoided and it can be ensured that customers are classified and treated in the same manner for all business processes implemented. Altogether, look-ahead fosters such a reuse of business rules; as a prerequisite, business rules must be easy to find and be described in a way comprehensible to domain experts.

General Requirements for Look-Ahead

Generally, look-ahead presumes that existing technical specifications can be easily found by business process designers. For this purpose, a high-level description understandable for end-users of the respective business domain is required. How this description should look like depends on the respective process aspect. Besides these differences, look-ahead should meet a number of general requirements as well:

- For each process aspect requiring look-ahead during business process design, techniques for describing technical artifacts (e.g. service or business rule) at a certain level of abstraction are required. Appropriate descriptions are crucial when considering the fact that the selection of a particular artifact might influence the design of business process models (i.e. their structure). Furthermore, it must not be ambiguous for process implementers, which concrete technical artifact shall be (re-)used when implementing a specific step of the business process.
- The descriptions of the technical artifacts should be easy to understand for business process designers having only few IT skills. To enable them to find the required artifacts, advanced techniques for searching in repositories and for browsing within descriptions are required.
- For any description of a technical artifact, its information content shall be complete, i.e., sufficient to enable decisions of business process designers on whether or not the artifact is appropriate for implementing a particular process step.
- A business process model may refer to the technical artifacts that shall be used during process implementation. For business process steps, therefore, it must be unambiguously specified which technical artifact shall be used for implementing them. Furthermore, corresponding references shall be stable, i.e., the identifier of the referenced artifact must not change. Then, a transformation into a process implementation becomes possible by using the implementation techniques offered by the process engine and substituting this identifier by the referenced implementation object (i.e. technical artifact).

FRONT-LOADING AND LOOK-AHEAD FOR SELECTED PROCESS ASPECTS

In the previous section, we introduced the basic principles of front-loading and look-ahead for different process aspects. This section provides more detailed insights into concrete techniques enabling front-loading for the organizational aspect or – to be more precise – for actor assignments. Furthermore, look-ahead is illustrated for the service perspective.

Front-Loading for Actor Assignments

As explained in the context of Step 4 from Figure (and Figure 1 respectively), contemporary business process modeling tools allow defining actor assignments only at a rough and imprecise level of detail. However, this contradicts with the general requirements existing in the context of front-loading, e.g., ease

of use for domain experts and availability of the information required by process implementers. In the following sections, we present selected techniques enabling front-loading for actor assignments. We further show how process implementers may transform the information generated by these techniques into a technical process implementation.

Approaches Enabling Business Process Designers to Define Actor Assignments

We present four modeling techniques for capturing actor assignments. According to the sequence in which they are presented, they offer increasing information content and functionality to users, while at the same time requiring extended support by business process modeling tools.

Approach 1 (Free Textual Description of Actor Assignments)

A simple modeling technique that can be realized in contemporary business process modeling tools is to describe the actor assignment of a process step in terms of free text. For capturing the respective text, annotation objects (Figure 2), process step attributes, or attached documents may be used depending on the given business process modeling tool. For example, in Figure , the following text might be assigned to Step 4: “The actor shall possess role ‘developer’ and belong to the same department as the change requestor.” In principle, with this simple technique, a good information quality can be achieved. Furthermore, process designers do not need to have any specific competencies to apply this modeling technique.

As a drawback, however, the quality of the resulting actor assignment descriptions cannot be ensured, i.e., the textual descriptions might be imprecise or ambiguous. In particular, process designers will be unable to guess and hence to use exactly the same names in their free text descriptions as the ones defined for OrgObjects (e.g. roles) in the organizational model of the enterprise. Regarding the above example (i.e. the actor assignment of Step 4), the provided free text refers to role developer instead of hardware developer. During process implementation, it has to be guessed (or inquired), which organizational object shall be used in fact. Furthermore, any combination of the different parts of an actor assignment (using AND / OR and respective priorities or brackets) might not be unambiguous. Finally, manual interpretations by the process implementer require mental effort.

Approach 2 (Semantic Description of Actor Assignments)

An extension of Approach 1 is to restrict the names of OrgObjects to a pre-defined vocabulary, i.e., to use an ontology. For example, when assigning a role name to a human task, the business process designer shall be forced to select this name from a pre-defined list. If the respective names of OrgObjects reference existing IT artifacts, this approach realizes look-ahead to a certain degree. A more advanced variant of this approach is to analyze the created descriptions automatically in order to detect references to OrgObjects. For this purpose, it must be checked whether the names used for OrgObjects are contained in the list of pre-defined names. In the example introduced in the context of Approach 1, for instance, such an analysis might reveal that the word “developer” shall correspond to a role name. Furthermore, a subsequent comparison with the pre-defined vocabulary might detect that this vocabulary does not contain role name “developer”, but role names “software developer” and “hardware developer”.

In such a scenario, the business process modeling tool shall ask the process designer to state the respective actor assignment more precisely. Finally, the business process designer will then substitute role name “developer” with “hardware developer”. Overall, Approach 2 results in more precise descriptions compared to Approach 1 since the names of OrgObjects become unambiguous. In turn, this reduces the effort for process implementers. As for Approach 1, however, costly analyses and interpretation of the arbitrarily structured parts of the textual actor descriptions are still necessary.

Approach 3 (Template-based Description of Actor Assignments)

The drawbacks discussed in the context of Approaches 1 and 2 are resolved by Approach 3. Not only the names of OrgObjects, but also the other textual parts of an actor assignment are now based on pre-defined artifacts. To be more precise, a set of templates covering all relevant types of actor assignments is provided to business process designers. Figure depicts examples of such templates.

Template Name	Description
Role(x)	Potential actors must possess role x (e.g. x = 'hardware developer')
Department(x)	Potential actors must be members of department x
Competence (x)	Potential actors must have competence x (e.g. x = 'knowledge of German')
Group(x)	Potential actors must be a member of group x (e.g. x = 'Project Passenger Safety')
Supervisor(x)	Potential actors are supervisor of person x
RoleAndDept(x,y)	Potential actors have role x and are members of department y

Figure 5. *Examples of template-based descriptions of actor assignments.*

To define an actor assignment of a process step, the business process designer selects a template and provides the names of the concrete OrgObjects that shall replace the template parameters (e.g. x or y in Figure). In order to avoid ambiguities, again, these names shall originate from a pre-defined vocabulary. Regarding Step 4 of our running example, for instance, template RoleAndDept(x, y) of Figure will be selected. Furthermore, the following OrgObjects will be assigned to the parameters of this template: x = "hardware developer" and y = "department of actor working on process step 'creation of the change request'". Overall, Approach 3 results in rather precise actor assignments. Only references to preceding process steps or process variables, as required in the context of dependent actor assignments, might be ambiguous. However, this can be improved with appropriate support of the business process modeling tool, e.g., only references to existing attributes (e.g. actor) of a process step shall be allowed.

In summary, process implementation efforts can be significantly reduced when using Approach 3. As a drawback, however, more complex actor assignments can only be expressed if a high number of specific templates considering all possible combinations of OrgObjects is provided (e.g. RoleAndDept, RoleAnd-Competence, DeptAndGroup, and so forth).

Approach 4 (Using Combinable Templates to Describe Actor Assignments)

To cope with the combinatorial problem when pre-defining actor templates, Approach 4 only pre-defines elementary templates like Role(x) or Department(x). However, these may be combined by the business process designer to more complex actor assignments using Boolean operators (i.e. AND, OR, NOT). To which extent a business process designer will be able to define such combinations, however, depends on his mathematical skill level. Generally, one cannot expect that a domain expert will be able to correctly define Boolean expressions without any tool support. Therefore, any business process modeling tool should provide sophisticated support for defining actor assignments.

First, it must allow for the selection of elementary templates (Figure 3a), the assignment of object names from a pre-defined vocabulary to template parameters, and the definition of references to process steps and process variables. Second, a business process modeling tool must allow combining the resulting actor assignments to more complex ones based on a collection of pre-defined and comprehensible combination types. The semantics of the latter must be appropriately explained to business process designers (see Figure 3b for an example), i.e., it will not be sufficient to present only a list of Boolean operators without further explanation.

Transforming Actor Assignments of the Business Level to the Process Implementation

The actor assignments defined by domain experts for the different steps of a business process model (see upper part of Figure 1) must be transformed into technical actor assignments at the system process level (see lower part of Figure 1). We will shortly discuss how a process implementer may perform such a transformation as well as the method that may be used to identify the concerned step(s) of the system process.

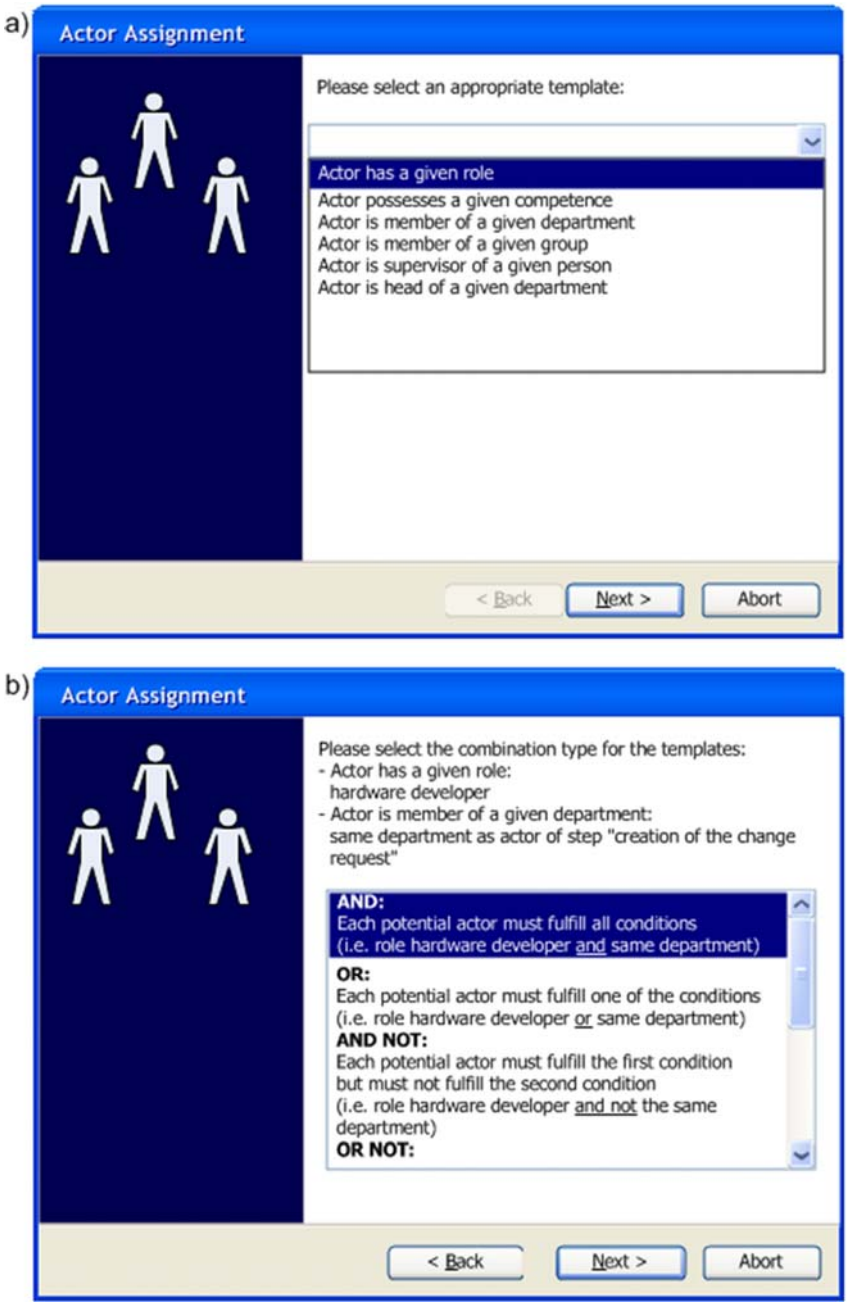

Figure 3. *Wizard for a) selecting an elementary template and b) combining templates.*

Deriving Technical Actor Assignments

We have presented four approaches enabling front-loading of actor assignments. For all of them, technical actor assignments may be manually derived by the process implementer. In this context, the information made available through front-loading significantly reduces process implementation efforts. Furthermore, the efforts for analyzing this information (and perhaps for contacting domain experts for further clarification) decrease from Approach 1 to Approach 4. Furthermore, Approaches 3 and 4 allow for a semi-automatic generation of technical actor assignments. As described, Approach 3 suggests using exactly one template for describing the actor assignment of a particular process step in a business process model. When providing standard implementations for each of these business level templates, the process implementation tool will be able to automatically select the right implementation of an actor assignment for system process steps (see lower part of Figure 1).

Furthermore, if the names of OrgObjects are selected from a pre-defined vocabulary at the business process modeling level, the references to corresponding technical OrgObjects can be automatically generated as well. The only remaining ambiguous part concerns the handling of references to preceding steps within actor assignments (e.g. “actor of process step B shall belong to the same department as the actor of process step A”). Regarding such dependent actor assignments, the process implementer must transform the respective part of the information provided by the business process model manually.

In principle, the same procedure can be applied in the context of Approach 4, but with an important extension: the transformation tool generating the technical actor assignments for a system process step now has to consider that the respective business level specification may consist of several elementary templates (i.e. basic actor assignments) combined with Boolean operators. In particular, the transformation tool must check whether the used process engine actually supports the technical actor assignment derived from this.

Handling Mapping Types

When deriving technical actor assignments, another relevant issue concerns the relation between the steps of a business process model and the ones of its related system process model (Figure 1) – note that the latter provides the technical basis for any process implementation. While for certain process steps there exists a 1:1 mapping between these two levels, in other cases a process step from a business process model is split into several steps of the system process model or vice versa, i.e., several steps from the business process model are merged to one system process step (see Figure 1 for examples).

Figure 1 shows that the BIMM that we described in the background section, is useful for deriving the system process step(s) corresponding to a particular step of the business process model. For example, regarding business process model step C in Figure 1, to which mapping type “Rename” is applied, it is obvious that the corresponding technical actor assignment must be defined for system process step Y. In particular, this step can be identified automatically when implementing the process. In turn, mapping type “Split” is more difficult to handle for process implementers. Consider again Figure 1, where business process step A is realized by the two system process steps T and U. Regarding the actor assignment of step A, the process implementer then has to figure out which of the two system process steps shall be considered.

Basically, there exist several options for dealing with such splits:

1. The actor assignment of the business process model is used to generate the technical actor assignments for both system process steps independently, e.g., by assigning the same role to these two steps. However, then different actors (with same role) might perform the two steps.

2. The actor assignment of the business process model is used for generating the technical actor assignment of system process step T. For system process step U, the actor assignment 'Same performer as for step T' is chosen to ensure that both steps are performed by the same actor.
3. The actor assignment of the business process model shall be applied either to T or U if only one of these two steps constitutes a human task. Note that technical actor assignments are only required in the context of human tasks, but not for automatically performed system process steps (e.g. Web Service calls). When using mapping type "Split", in many cases, only one of the derived system process steps actually corresponds to a human task, while the other steps are automated ones.

Concerning mapping type "Merge", the corresponding system process step can be always identified unambiguously. However, since several steps of the business process model are merged to one system process step (e.g. in Figure 1, steps D and E are merged resulting in system process step Z), it should be checked whether there exist contradictory actor assignments of these different business process steps – this would be an error in the business process model. Regarding mapping type "Insert" (e.g. step X in Figure 1), usually, no technical actor assignment is required since this mapping type is solely used for technical activities executed without any user interaction. Finally, in the context of mapping type "Delete" (i.e. a process step of the business process model is not mapped to any step of the system process model), no transformation is required at all.

Note that the BIMM is not only useful when implementing a system process, but also in the context of later business process changes. For example, assume that a domain expert decides to change an actor assignment of a business process model during its implementation phase or even after deploying the IT artifacts of the implemented process. Based on the BIMM, it then will be easy to figure out, which system process steps are affected by this change. Furthermore, when taking the information provided by front-loading into account, changes of actor assignments at the business process level can be quickly and flexibly transformed into adaptations of the corresponding technical actor assignment. In turn, this increases both process flexibility and process quality in Service-driven environments.

Look-ahead for Services

Taking the service aspect (i.e. calling a service to execute a process step), we exemplarily discuss how look-ahead can be realized. Firstly, we discuss selected approaches for publishing information about existing services in a manner comprehensible to business process designers. Secondly, we explain how service specifications, created by business process designers, can be used during process implementation.

Approaches for Publishing Services for Business Process Designers

We present three approaches for publishing information about existing services in a manner comprehensible to business process designers and enabling them to use this information in the context of look-ahead. The three approaches mainly differ in respect to the information source they use: a technical ***SOA repository***, ***business repository***, or business process modeling tool. Since each of the three approaches meets the requirements for service look-ahead, the approach best suited depends on the IT landscape and the business processes to be supported.

Approach 1 (Extending the Technical SOA Repository)

In many companies, for already implemented services, there exist technical descriptions (e.g. WSDL files) usually stored in a SOA repository. Obviously, such SOA repositories might serve as the basis for service look-ahead as well. However, the technical information provided by them is hardly comprehensible to business process designers. Therefore, additional information should be maintained to enable service look-ahead for business process designers as well. For this purpose, attributes capturing

business information about implemented services should be added to the meta-model of the SOA repository. Examples of such attributes include textual service descriptions from a business perspective (i.e. the business function supported by the service), service category (e.g. category “data delivery” for the service used by Step 5), descriptions of the input / output data objects of the service (e.g. “part data that contains the attributes …”), the domain or IT system providing the service (e.g. “product data management in the context of vehicle development”), and contact person. Respective attributes can be used by business process designers in the context of service look-ahead, i.e., when searching for a service (e.g. based on category and domain) and deciding about whether a specific service matches the requirements of the given business process step.

Usually, it is possible to extend the meta-model of a (technical) SOA repository. Respective options are provided by SOA tools like WebSphere Service Registry and Repository (WSRR) (IBM, 2007) and CentraSite (Schneider &Vaughan-Brown, 2008). Extending a meta-model means that additionally required attributes may be created and added to the already existing SOA repository of an enterprise. However, note that the quality of Approach 1 does not only depend on the extension features of a SOA repository, but also on the definition and maintenance of complete and consistent service descriptions, i.e., information quality is crucial. The latter should be controlled by appropriate governance processes that ensure both completeness and comprehensibility of the service information provided. In particular, passing respective checks should be a prerequisite for releasing any technical service implementation.

Approach 2 (Business Repository)

As an alternative to Approach 1, the service descriptions comprehensible to business process designers can be stored in a separate business repository. In particular, a description of a business service may be also created if the specification of the respective technical (i.e. implemented) service is available, i.e., a corresponding entry in the SOA repository exists. Note that such a late business service specification enables domain experts to decide for any registered technical service whether it is meaningful from a business perspective, i.e., whether a corresponding entry shall be created for the business repository as well. The information to be maintained in a business repository is similar to the one covered by the additional attributes mentioned in Approach 1. Since a business repository focuses on the business aspect, however, it may comprise additional attributes and details.

In principle, a business repository does not contain any technical information about services (e.g. WSDL descriptions). However, service look-ahead must be able to determine the technical service (i.e. the implemented service) to be invoked when a particular step of the system process becomes enabled. For this purpose, each business service must be associated with a technical service, e.g., by adding references to technical services from the SOA repository to entries of the business repository. Alternatively, technical services from the SOA repository can be enriched with references to corresponding business services.

Offering a separate business repository relieves business process designers from browsing through technical SOA repositories when searching for a particular business service. Furthermore, the meta-model schema of a business repository meets the needs of business users, and does not have to take the requirements of IT departments, using the same repository (as in Approach 1), into account. Again, completeness and quality of the information captured in a business repository (e.g. service descriptions) must be ensured by establishing appropriate governance processes. Due to the co-existence of business repository and technical SOA repository, it must be ensured that the release of a technical service will be accompanied by the definition of a corresponding business service where appropriate.

Approach 3 (Graphical Model in a Business Process Modeling Tool)

Business process designers use high-level modeling tools for creating and changing business process models. Usually, a business process modeling tool represents its models in a graphical notation (e.g. EPC, BPMN). In particular, all models created with such a tool belong to the business level. Therefore, a business process modeling tool represents an appropriate source for business service descriptions as well. In particular, for business process designers, it will be most convenient if they are able to define or access business service specifications within the same tool.

To realize this approach, a business process modeling tool must allow creating and accessing descriptions of business services. As for other model components (e.g. process model, data model, and organizational model), a graphical notation should be provided for this, i.e., a special diagram type for modeling business services is required. As illustrated by Figure 4, the central element of such a diagram type is the business service itself. All relevant properties can then be described by objects connected to the respective business service. Taking the annotations provided in the context of Figure 4, one obtains a concrete example of the service required for realizing Step 5 in Figure .

More precisely, the business service has name “provide part data”, its input (output) data object is “part number” (“part data”), it is provided by organizational unit “product data management”, and so forth.[4] Each of these objects is at least described by its name. In addition, a textual description (i.e. a document) or another diagram may be assigned to the object. For example, a document attached to the organizational unit “product data management” may describe that this unit “belongs to the domain vehicle development, centrally manages the PDM data for all business units, and realizes the operations of the PDM system”. Furthermore, the output data object “part data” may be detailed through a business object diagram, which contains all attributes of data object “part data”, e.g., name, weight, procurement costs, and responsible developer of the part.

As described in (Stein, 2009), ARIS supports ***Service Allocation Diagrams***, which may be used to specify business services as well. Another object type that may be connected with business service objects to describe the corresponding business functions is “capability”. Regarding the service example from Figure 4, related capabilities may describe that “part data is returned”, “part is found by part number”, “part data is delivered even for unreleased parts”, and so forth. Finally, respective capabilities can be also used by business process designers to find an appropriate service in the context of service look-ahead.

Object type “software service type” (Figure 4), represents the technical service corresponding to business service “provide part data”. For example, this object may contain the WSDL description of the technical service or refer to it, i.e., it is not intended for business process designers, but for process implementers.

Comparing the Approaches

As major advantage of Approach 1, all relevant information is stored within one and the same SOA repository, i.e., both the technical and the business perspective are maintained in the same repository. Hence, the two perspectives can be easily kept consistent since no references have to be maintained across different tools. Furthermore, Approach 1 can be realized with moderate effort, especially if a technical SOA repository already exists. As a drawback, however, business process designers have to work with a repository that still is based on a rather technical structure.

[4] Respect that it is not necessary to use all (possible) object types depicted in Figure 7 for describing a specific business service. Only such objects shall be connected to a service type that really contributes to explain the purpose and function of this service.

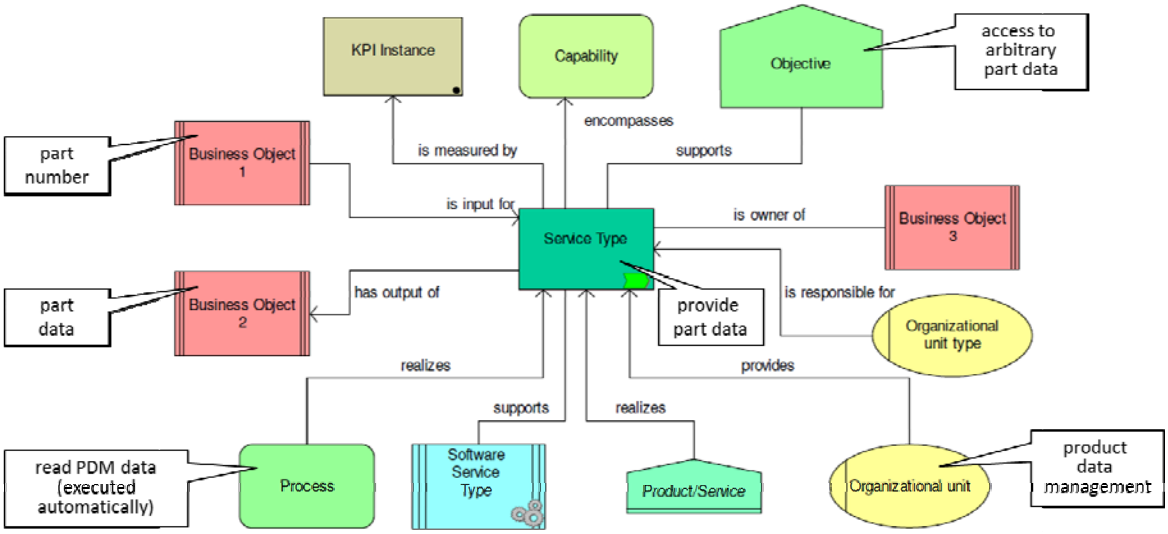

Figure 4. *The service allocation diagram of ARIS used for describing a (business) service type.*

Approach 2 avoids these disadvantages since business process designers do not work with a technical repository. In particular, the meta-model of the business repository may respect the specific requirements of business users without any restriction, e.g., it becomes possible to include additional business-related information, if required. As a drawback, Approach 2 requires additional realization efforts as well as efforts to keep technical and business information consistent across the two repository systems.

As major advantage of Approach 3, the graphical user interface of a modeling tool that the business process designer is familiar with is used. However, only object types offered by the respective modeling tool can be used (or the tool must be extended which might require high customization costs). Furthermore, respective information should be made accessible to all business process designers even if they are assigned to different development projects.

Realizing Service Calls in the Context of Process Implementation

When using Approach 1, the business process model directly refers to technical services. By contrast, when using Approaches 2 and 3, a business service is referenced instead. In turn, this business service refers to a unique technical service. By using the information provided by look-ahead, for process implementers, it becomes obvious which technical service shall be used for implementing a particular step in a business process model.

The presented BIMM (Figure 1) enables determining the system process step belonging to a given business process step. As discussed in the context of actor assignments, however, multiple steps of a system process may belong to the same business process step (i.e. mapping type “Split” is used). Even for this case, however, the respective step of the system process can be identified unambiguously if only one of these steps is realized through an automated service call and the other steps correspond to human tasks (e.g. consider business process step A and the two system process steps related to it, i.e., human task T and service call U). Only if several technical services and automated steps are assigned to the same business step, ambiguities might occur, requiring further analyses or inquiries.

Again, the BIMM is very useful when business requirements for already existing process implementations change. For example, the business division might decide that a different service shall be

used for Step 5 in Figure going forward, e.g., if a new BoM system is released offering more appropriate part data than the PDM system does. In such a situation, it is easy to adapt the process implementation. First, look-ahead allows deriving the required technical service unambiguously. Second, the existing BIMM allows deriving the system process steps affected by the change.

CONCLUSION AND OUTLOOK

In a Service-driven environment, business process design necessitates advanced techniques enabling front-loading and look-ahead. Both are crucial to realize process implementations that actually meet business requirements on one hand, but do not ignore the technical SOA environment existing in an enterprise on the other. The chapter analyzed the process aspects that benefit from front-loading and look-ahead and discussed general requirements for these two fundamental concepts. To provide detailed insights into how respective techniques can be transferred to practice, front-loading techniques were illustrated for the aspect of actor assignments. Thereby, focus was not on developing new modeling techniques, but to specify relevant information for implementing processes as early as possible during the process design phase. Furthermore, basic look-ahead concepts were illustrated for the aspect of services. A particular challenge in this context is to make existing technical services known to business process designers.

We explored different process aspects that may benefit from front-loading and look-ahead in general. However, business process designers must be also able to use the resulting methodology. In particular, they should not be burdened by the introduction of a large number of new techniques. Furthermore, front-loading and look-ahead should only be used for aspects not overextending the technical capabilities of business process designers (note that it depends on the skill level of a business process designer which aspects actually lead to an overextension). This means, in a PAIS development project, it has to be considered which process aspects are most relevant for front-loading and look-ahead and for which aspects it is realistic to use the techniques proposed. For the other aspects, the required information may be determined at later project phases or decisions can be made autonomously during the technical specification or implementation. Such a limited approach may be acceptable for certain process aspects or detail levels in order to keep business process design as easy as required.

The proposed approaches still have to be tested in real implementation projects in order to prove that they are applicable in the large scale and really result in significant benefits. The mentioned aspects and techniques may be used as a basis (similar to a checklist) for selecting the front-loading and/or look-ahead aspects that shall be included into the methodology of the given development project. It is reasonable to base this selection on experiences made during previous development projects. Using front-loading or look-ahead is recommended for process aspects whose clarification during process implementation resulted in considerable delays or whose transformation to the technical level caused erroneous process implementations due to missing details in business process models.

REFERENCES

Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Ganapathy, S., & Holley, K. (2008). SOMA - a method for developing service-oriented solutions. *IBM Systems Journal* 47:377-396

Bauer, T. (2009). Substitution rules for task performers in process-oriented applications. *Datenbank-Spektrum*, 9(31)*:*40-51 (in German).

Bodenstaff, L., Wombacher, A., Reichert, M., & Jaeger, M.C. (2008). Monitoring dependencies for SLAs: the MoDe4SLA approach. *Proc IEEE 5th Int'l Conference on Services Computing (SCC 2008)*, Honolulu, Hawaii, USA, July 2008, IEEE Computer Society Press, pp. 21-29.

Buchwald, S., Bauer, T., & Reichert, M. (2012). Bridging the gap between business process models and service composition specifications. In: Lee et al. (Eds.). *Service Life Cycle Tools and Technologies: Methods, Trends, and Advances* (pp. 124–153). IGI Global.

Buchwald, S. (2012). *Increasing consistency and flexibility of process-oriented applications by service-orientation.* PhD thesis, University of Ulm, Germany (in German).

Chen, H.M. (2008). Towards service engineering: service orientation and business-IT alignment. *Proc. 41st Hawaii Int. Conf. on System Sciences*, IEEE Computer Society.

Dadam, P., & Reichert, M. (2009). The ADEPT project: a decade of research and development for robust and flexible process support - challenges and achievements. *Computer Science - Research and Development*, 23(2): 81-97, Springer.

Deeg, M. (2007). SOA starts far beyond BPEL – service-oriented business process modeling as basis for a SOA. *OBJEKTspektrum - Onlineausgabe* (in German).

Enderle, R. (2009). *Early modeling of process aspects relevant for process execution at a business level – process modeling in a SOA.* Master thesis, University of Ulm (in German).

Engels, G., Hess, A., Humm, B., Juwig, O., Lohmann, M., Richter, J.P., Voss, M., & Willkomm, J. (2008a). A method for engineering a true service-oriented architecture. *Proc 10th Int Conf. on Enterprise Information Systems*, Barcelona, 272-281.

Engels, G., Hess, A., Humm, B., Juwig, O., Lohmann, M., Richter, J.P., Voss, M., & Willkomm, J. (2008b). *Quasar enterprise: service-oriented design of applications landscapes.* Dpunkt-Verlag.

Erl, T. (2005). *Service-oriented architecture: concepts, technology, and design*. Prentice Hall.

IBM (2007). *WebSphere service registry and repository handbook*. IBM Redbook.

IDS Scheer (2005). *Business process management: ARIS value engineering – concept.* White Paper.

Kolb, J., Hübner, P., & Reichert, M. (2012). Automatically generating and updating user interface components in process-aware information systems. In: *Proc. 20th Int'l Conf. on Coop Information Systems (CoopIS'12)*, Rome, Italy, LNCS 7565, Springer.

Künzle, V., & Reichert, M. (2009). Integrating users in object-aware process management systems: issues and challenges. *Proc. BPM'09 Workshops, 5th Int Workshop on Business Process Design (BPD'09)*, September 2009, LNBIP 43, Springer, pp. 29-41.

Künzle, V., & Reichert, M. (2011) PHILharmonicFlows: towards a framework for object-aware process management. *Journal of Software Maintenance and Evolution: Research and Practice*, 23(4): 205-244, Wiley.

Künzle, V., Weber, B., & Reichert, M. (2011). Object-aware Business Processes: Fundamental Requirements and their Support in Existing Approaches. *International Journal of Information System Modeling and Design (IJISMD)*, 2(2): 19-46, IGI Global.

Lanz, A., Weber, B., & Reichert, M. (2013). Time patterns for process-aware information systems. *Requirements Engineering Journal*, Springer, 10.1007/s00766-012-0162-3.

Lanz, A., Weber, B., & Reichert, M. (2010). Workflow time patterns for process-aware information systems. *Proc Enterprise, Business-Process, and Information Systems Modelling: 11th International Workshop BPMDS and 15th International Conference EMMSAD at CAiSE'10*, Hammamet, Tunisia, June 7-8, 2010, LNBIP 50, Springer, pp. 94-107.

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.

Offermann, P. (2008). SOAM – a method to concept enterprise software with a service-oriented architecture. *Wirtschaftsinformatik.* 6: 461-471 (in German).

Offermann, P., & Bub, A. (2009). A method for information systems development according to SOA. *Proc 15th Americas Conf. on Information Systems,* San Francisco.

Reichert, M., & Dadam, P. (1997). A framework for dynamic changes in workflow management systems. *Proc. 8th Int'l Workshop on Database and Expert Systems Applications,* Toulouse, France, September 1997, pp. 42-48.

Reichert, M., & Dadam, P. (1998). ADEPTflex - supporting dynamic changes of workflows without losing control. *Journal of Intelligent Information Systems*, 10(2): 93-129.

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

Reichert, M. et al. (2009). Enabling Poka-Yoke workflows with the AristaFlow BPM Suite. *Proc. BPM'09 Demonstration Track*, Ulm, Germany, September 2009, CEUR Workshop Proceedings 489.

Reichert, M., Rinderle-Ma, S., & Dadam, P. (2009). Flexibility in process-aware information systems. *LNCS Transactions on Petri Nets and Other Models of Concurrency (ToPNoC)*, Special Issue on Concurrency in Process-aware Information Systems, LNCS 5460, Springer, Vol. 2, pp. 115-135.

Reichert, M., & Weber B. (2012). *Enabling flexibility in process-aware information systems: challenges, methods, technologies*. Springer.

Rinderle, S., & Reichert, M. (2005). On the controlled evolution of access rules in cooperative information systems. *Proc 13th Int'l Conf. on Cooperative Information Systems (CoopIS'05)*, Agia Napa, Cyprus, LNCS 3760, Springer, pp. 238-255.

Rinderle-Ma, S., & Reichert, M. (2007). A formal framework for adaptive access control models. *Journal on Data Semantics* IX, Springer, LNCS 4, pp. 82-112.

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.

Sadiq S., Sadiq W., & Orlowska M. (2005). A framework for constraint specification and validation in flexible workflows. *Information Systems* 30(5):349 – 378.

Schneider, G., & Vaughan-Brown, J. (2008). The CentraSite community - fast-tracking SOA governance using best-of-breed solutions. White paper, Software AG.

Stein, S., Kühne, S., Drawehn, J., Feja, S., & Rotzoll, W. (2008). Evaluation of OrViA framework for model-driven SOA implementations: an industrial case study. *Proc 6th Int Conf on Business Process Management*, Milan, 310-325.

Stein, S. (2009). *Modelling method extension for service-oriented business process management.* PhD thesis, University of Kiel, Germany.

Shuster, L. (2008). Project-oriented SOA. *SOA Magazine, XXI.*

Weber, I., Hoffmann, J., Mendling, J., & Nitzsche, J. (2007). Towards a methodology for semantic business process modeling and configuration. *Proc. 2nd Int'l Workshop on Business Oriented Aspects concerning Semantics and Methodologies in Service-oriented Computing*, 176-187.

Weber, B., Reichert, M., Wild, W., & Rinderle-Ma, S. (2009). Providing integrated life cycle support in process-aware information systems. *Int'l Journal of Cooperative Information Systems*, 18(1): 115-165.

Weber, B., Reichert, M., & Rinderle-Ma, S. (2008). Change patterns and change support features - enhancing flexibility in process-aware information systems. *Data and Knowledge Engineering*, 66(3): 438-466, Elsevier

Weber, B., Rinderle, S., & Reichert, M. (2007). Change patterns and change support features in process-aware information systems. *Proc. CAiSE 2007*, pp. 574-588, Springer.

Weber, B., Sadiq, S., & Reichert, M. (2009). Beyond rigidity - dynamic process lifecycle support: a survey on dynamic changes in process-aware information systems. *Computer Science - Research & Development*, 23(2), 47-65, Springer.

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