Service-based Processes – Design for Business and Technology

Martin Henkel
Stockholm University and
Royal Institute of Technology
Forum 100, 164 40 Kista
Sweden
+46 8 16 16 42
martinh@dsv.su.se

Jelena Zdravkovic
University of Gävle and
Royal Institute of Technology
801 76 Gävle
Sweden
+46 26 64 83 06
jzc@dsv.su.se

Paul Johannesson
Stockholm University and
Royal Institute of Technology
Forum 100, 164 40 Kista
Sweden
+46 8 16 16 71
pajo@dsv.su.se

ABSTRACT
Composition of software services is a fundamental part in supporting enterprise business processes. Designed properly, executable processes can be used to closely support business processes by the integration of existing software services. In order to support business processes the design of the executable process must closely follow the business events and activities, as perceived by business actors. However, the design must also consider technical issues such as limitations in existing technology and systems. In this paper we examine how technical system constraints influence the realization of business processes. Based on this examination we present a set of realization types that describes the transformation from a business process into its realization as an executable process. We also propose design criteria that need to be adhered to in order to cater to both business and technical needs.

Categories and Subject Descriptors
H.1.0 [**Models and Principles**]: General*;*
H.4.1 [**Information Systems Applications**]: Office Automation – *Workflow management*

General Terms
Design, Languages. Theory.

Keywords
Business processes, Executable processes, Service coordination.

1. INTRODUCTION
Large scale service-oriented systems rely on complex message exchanges between individual software services. As the complexity and the number of interactions increases, there is a need to explicitly control and implement the service interactions. Executable processes enable individual services to be composed to support new, complex business interactions [1, 2].

When designing executable processes, consideration must be paid to both business requirements, and the technical context that the process should be executed in. When concentrating on the business requirements the business process is modelled to closely resemble the business activities and message exchanges. Adding technical constraints to the design process means that the designed executable process needs to be aligned with existing software services. For instance, limited system functionality in existing systems can affect the design of an executable process. Today's vast amount of legacy systems can even result in a design shift where technical aspects to some extent come into focus of the process design. This is especially evident if an executable process language is used to solve system integration issues. The coordination of message exchanges between systems will then be a central aspect of the process.

In this paper, we discuss differences between processes modelled from the pure business perspective and those modelled with considerations of technical aspects.

When designing a process from the *business* perspective, the focus is to "automate the business", i.e. to solve problems that are expressed in business terms. Thus, the aim of the design is to model and later on to implement business processes that capture the message exchanges, activities and roles that are part of a business. For instance, parallel activities in a process will be utilized to denote concurrent business operations. Another example is that a sequence of activities with specific messages to exchange will be used to steer business behaviour according to business contracts. Such processes have the advantage of being easy to understand for people within the business. Another advantage is that the close mapping between the business and the process makes it possible to easily rearrange the activities and sequences when the business changes. Workflows [3] are an example of systems that closely depict the business process as understood by people engaged in the business.

ICSOC'04, November 15–19, 2004, New York, New York, USA.

When designing processes from the *technical* perspective, the focus is to leverage business support by utilizing existing software services. This requires for solving system integration as well as synchronization problems. Just as with modelling processes from the business perspective, the final goal is to support the message exchanges and activities of the business. However, this common goal is reached by integration of existing services. Thus, when constructing technical processes the designer must deal with both business behaviour and the behaviour of existing services. The modelled process will at least partly describe the behaviour of the existing system. For example, it might be decided that parallel activities must be used because the same information is required to be sent to two back-end services. Another example is that a set of alternative activities will be used to enable choice of a system used for communicating a message. These concepts are considerable more "technically" oriented compared to those dealt with when designing pure business processes. Middleware solutions that utilise integration patterns [4] such as message routing, splitting and joining to interconnect back-end systems] are an example of where process descriptions partly deal with system issues rather that business issues. By designing processes from the technical perspective it is possible to coordinate systems interactions in a straightforward way.

In order to build service-based systems that cater to both business needs and technical constraints, it is important to be aware of how a business process, when realized, is affected by existing services. It is also crucial to apply architectures that adhere to the business needs, and at the same time solves technical issues.

In this paper we utilize a process description framework to highlight the differences of processes modelled from the business and technical perspectives. By using the framework, we identify a set of *realization types* that describes possible transformations from a business process into a more technically oriented process. Furthermore, based on the realization types and a notion of lossless realization, we examine under which conditions business and technically oriented processes can coexist in a layered architecture.

The problem of non-fitness between enterprise business processes and information system functionality has been extensively studied in the research community [5, 6, 7, 8, 9]. Our work differs in two aspects. First, we discuss misfit between business and technical perspectives in process modelling, and with focus on executable model specifications. Second, we concentrate our research towards bridging the gap between existing business- and technology-dependent model layers, by introducing a set of transformation patterns. Our usage and comparison of two model layers is similar to the model layers and model transformations in the Object Management Groups (OMG) Model-Driven Architecture (MDA) approach. MDA is a general approach for separating models from the underlying platform technologies [10]. Although the ideas of MDA can be applied to create automated model transformations, for example for component systems [11], MDA is rather a framework for such transformations. Thus, the realization types/transformations that we present in this paper can be used within the MDA framework.

This paper is structured as follows. In the next section we define business and technical processes. Furthermore, we discuss the notion of a technical process as a realization of a business process. In Section 3 an example of a business process is presented, as well as how it is realized following a set of system constraints. In Section 4, we describe a framework that can be utilised to analyze the design and constituents of executable processes. This framework is applied in Section 5 to analyze the possible realization transformations that can be utilized to construct a technical process. Section 6 examines the requirements that need to be followed if a business process and a technical process should co-exist in the same architecture. The paper ends with a discussion of the presented results.

2. BUSINESS AND TECHNICAL PROCESSES

Analysis of a business might result in the design of one or more executable processes. These designs might later be implemented by using an executable process language, such as BPEL4WS [12] or YAWL [13]. Based on the discussion presented in Section 1, we distinguish between two types of process designs:

Business processes (B) are designed based on business concepts such as business events, activities and business actors. This means that the process reflects the pure business perspective.

Technical processes (T) are software realizations of business processes that are designed upon the business concepts and in accordance to existing systems and services. This means the technical process reflects both business perspective and technology concepts such as software services, applications and middleware products.

Given the definitions above, it can be said that technical processes realize business processes by using existing systems. Thus we have the relation that T belongs to the set R(S,B), where R is the function of realization, S are existing system and technology limitations, and T and B are processes that fulfil the same business goals. Since the function R(S,B) can result in several technical processes for the same input (S,B), we do not use the relation T=R(S,B).

If there are no constraints put on the realization from existing systems, the technical process will directly correspond to the business process (T=B). However, the realization can be affected by the following system limitations (S):

1. The business process cannot be implemented as-is, because of domain independent technology limitations, for example limitations in programming languages and communication protocols.
2. The existing domain specific software services, when composed in a process, do not match the documented business process.

Since a business process (as defined above) is defined in business terms, it has the advantage of being easy to understand and modify for people within the business. However, technical processes cannot be overlooked since limitations in existing systems are inevitable.

In the next section we give a concrete example of how system constraints can affect the realization of a business process.

3. EXAMPLE CASE

An example of a business process and its realization as a technical process is shown in Figure 1. The example, in the form of a technical process, is based on a case provided by Sandvik, a global industrial materials engineering company with offices in 130 countries. A major concern of Sandvik is integration and coordination of their existing ERP systems in the form of software services. For this coordination Sandvik utilizes Web service protocols, as well as middleware technology such as Microsoft Biztalk and IBM MQ. As one of the pioneers in the use of Web services, Sandvik has recognized the need to use executable business processes to handle the coordination of its services.

The business process in Figure 1(a) depicts a basic order process where the process is triggered by an incoming order request. Since we are using the Business Process Modelling Notation (BPMN) [14] we depict the customer with a swimlane/pool symbol, message events with encircled envelopes and message flows with dotted arrows. After an order confirmation is sent to the customer, the process is forked into two parallel flows. A shipment plan is constructed while the order is being processed. Later on, the flow is synchronized using an AND-join (called "AND gateway" in BPMN). Before the end of the process a notification that the product has been shipped is sent to the customer.

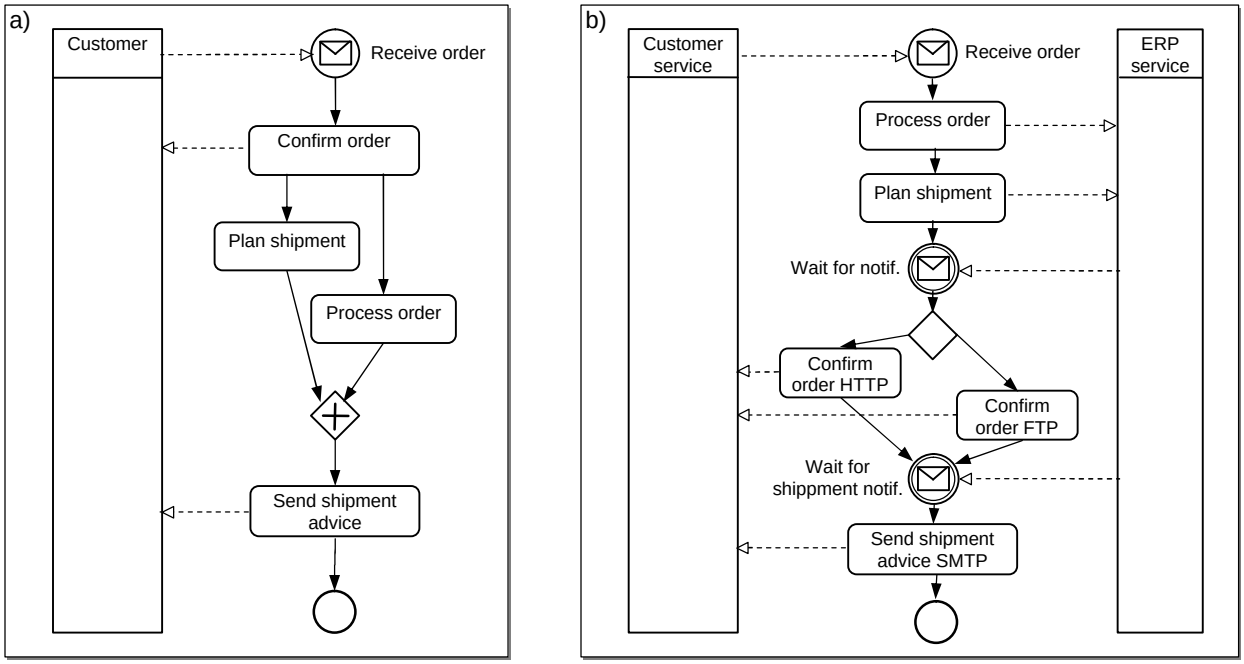

Figure 1. Business process (a) and realized technical process (b).

Figure 1(b) depicts the realized technical process. This process is an excerpt of the process supplied by Sandvik. The technical process is based on existing services, in this case the ERP system (depicted by the ERP service pool in Figure 1) and a service interface to the customers' information systems (the customer service pool in Figure 1). Compared to the business process, the technical process must adhere to a set of system constraints, S:

- S1 – The existing ERP service perform logistics planning and order processing in an integrated activity, a notification can be received when this process is completed.
- S2 – The validation of order information is integrated into the ERP service, order confirmation can be sent when the order is received by the ERP service.
- S3 – Based on the customers (software) service ability order confirmation should be sent as a HTTP message or a FTP file.

These system constraints affect the realization of the business process. The characteristics of the ERP system (S1) prompt us to put shipment planning and order processing in a sequence. Since the message event from the ERP system signals when the order confirmation can be sent (S2), the sending of the order confirmation is placed directly after (instead of before) shipment planning and order processing. Furthermore, the usage of different protocols for sending the order confirmation (S3) prompts us to use an XOR-split (called XOR gateway in BPMN) of the process flow.

Even though the example is small, it shows how a realization of a business process can be affected by system constraints. In this example we used a subset of the process supplied by Sandvik, for brevity reasons we excluded customer authentication and the handling of invoices. We also simplified the handling of protocols, the original process handles more protocols than FTP and HTTP for all activities that notify the customer.

The example elucidates the point that realization does change the design of a process. In the next section we define process design aspects to provide methodical examination of how different types of realizations affect the design of technical processes.

4. ASPECTS OF PROCESS DESIGN

In this section we introduce a conceptual framework to classify different aspects that constitute process design (specification). The framework is based on modeling aspects of workflows, as proposed in [15, 16].

The basic aspect of process specification is *functional*. The functional perspective describes how a process is decomposed, i.e. what activities are to be executed. Functionality of an activity is depicted by name, which uniquely identifies the goal of the activity; by message exchange, designated with input and/or output documents of the activity; and by constraints, which depict activity-internal constraints, such as pre-conditions and post-conditions. The functional aspect, thus, depicts only the semantics of activities. Implementation of activities, however, is not part of the process specification.

The *Behavioral* aspect depicts process control flow, i.e. when an activity is to be executed in relation to others. For specification of dependencies and coordination rules among activities, process specifications rely on a set of basic control flow constructs: sequence, parallel execution (AND split), synchronization (AND join) and conditional branching (OR/XOR split/join).

The *informational* aspect concerns process data. In a process specification, data are information concepts that are used as process attributes upon which flow rules are set and controlled, as well as information that the process exchanges with the external environment. An information concept is depicted by content and structure of contained data.

The *organizational* aspect depicts the distribution of responsibility of executing the activities. In order to model business of an organization, personal and technical resources have to be mapped to specific roles. Roles are related to activities they are intended (allowed) to execute. There may be different relations between roles and activities. For instance, it may be one or more roles allowed to perform an activity; and one role may be allowed to execute one or more activities in a process. By using roles it is possible to dedicate and control responsibilities of parties engaged in a process.

The *transactional* aspect concerns consistent execution and recovery of a set of activities. The design of the transactional aspect includes defining what to be considered as consistent states of the process, thus depicting particular transactional boundaries (scopes). In addition, process transactions may comply to different models, as they contain loosely coupled activities that may have short or long duration. The atomic transaction model [17] is used to administer a set of shorter activities, that all agree to enforce a common outcome by two-phase commit. In contrast, business (long-running) transaction model [18] rules more durable activities, where each activity enforces a globally visible outcome independently of the others; when an activity fails, the parent transaction rolls back to a consistent process state by compensating activities that had successfully completed.

In addition to the defined aspects, by the model of Jablonski and Rausch-Scott, workflows have to cope with other aspects, such as causal and historical (data logging). Those concepts are not, however, used in process specifications; they are related to the workflow environment and monitoring, which we do not consider.

In the next section we describe how each of the above process aspects may be affected when realizing a business process as a technical process.

5. PROCESS DESIGN AND REALIZATION

When designing a business as well as a technical process, each of the five process aspects must be considered. The input to the design is high-level business goals, as well as concrete technical and business requirements and constrains. As mentioned earlier, the input to the design of business processes and technical processes differ, and thus the end result differ.

In this section the primary business and technical design considerations for each of the process aspects is described. Furthermore, a set of *realization types* is identified for each aspect. Each realization type describes a possible transformation from a business to a technical process.

The total set of realization types covers the possible changes that need to be applied to a business process during realization. The selection of the realization types are based on the five process aspects as well as on differences in business and technical process design. The five aspects are used as a fundament to ensure coverage of the properties of processes, while the aspect definitions and the differences in business and technical process design form the basis for selecting realization types on a per-aspect basis.

5.1 Functional Aspect

When comparing the design of business and technical process from the functional perspective we compare how the functionality of the process is decomposed into activities.

Executable business processes cater directly to business needs describing the activities organizations should perform to achieve a common goal. Activities in a business process are modelled, therefore, according to activities performed in organisations and by humans.

When designing technical processes the design issues will shift towards expressing how systems and technology can be combined to solve business problems. In this case the activities are selected according to system operations. Thus, the functionality of existing systems will be the base for selecting the process activities.

Following the definition of the functional aspects in Section 4, activities in a business process could differ from those in the corresponding technical process, with respect to goals, message exchange, and/or imposed pre- and post-conditions. We argue, thus, that the functional aspect of a business process, may be designed in a corresponding technical process by the following realization types:

- Aggregation - an activity *a* in the business process, corresponds to more than one activity (a_1, a_2, a_3,.., a_n) in the technical process, where those activities jointly exchange the same messages as the activity *a*, thus achieving the same goal. As an example, the activity “process order” in the business process may correspond to activities “process order

header by system x" and "process order details by system y" in the technical process.

- Specialization - an activity *a* in the business process, corresponds to more than one activity (a_1, a_2, a_3,.., a_n) in the technical process, where each of those exchange the same messages as the activity *a*, but with different goals due to specialization to a particular system. For instance, as depicted in the example case in Section 3, "confirm order" activity in the business process reflects to the activities "confirm order by HTTP" and "confirm order by SMTP" in the technical process.
- Condition alteration – an activity a in the business process having similar goal and message exchange as an activity a_1 in the technical process, differs from activity a_1 in pre- and/or post-conditions. An example of a difference in pre-conditions that may occur is if an activity in the business process is defined to handle the currencies Euro and Dollar as valid input, while the corresponding activity in the technical process only supports Euro. In this case, the pre-condition of the technical activity is stronger than the corresponding business activity.

5.2 Behavioural Aspect

Comparing the design of the behavioural aspects of technical and business processes involves examining the criteria that governs flow order of activities.

In business processes, the rules that govern the flow coordination of the activities can be stated by simple business rules, such that payment should be done before a product is shipped. Thus, the design of the flow of activities must follow business contracts and other (implied or explicit) agreements of the business partners.

Technical processes must also adhere to the desired business behaviour. The order in which to perform activities might also be governed by limitations in the technical features of the back-end systems, for example by dependencies between existing services. A process that is designed in this fashion might have a quite different ordering of the activities, compared to a process designed from the business perspective only.

Based of the definition of the behavioural aspect in Section 4, and the above discussion, we identify the following realization types that can be applied to transform the behavioural aspect of a business process into a technical process:

- Reordering (concurrency) – sequentially ordered activities in one process (business or technical) may correspond to parallel ordered activities in the other process. For instance, limited transaction capabilities might result in a technical process where an activity that does not support transactions is placed last in a sequence of activities. This design can be placed in contrast to a business process, where the same activities might run in parallel.
- Reordering (sequencing) – activities ordered in a sequence in the one process may be, in the other process, sequenced differentially. For example, contract allows the payment to be done after delivery, but from the technological view, it cannot, as delivery system requires the payment number.
- Condition change - based on a conditional statement, the flow of a process might take different routes. Such a condition might have the same, fewer or more possible branches in a technical process compared to a business process. An example is that a technical process might introduce additional branches to distinguish protocol related issues, such as those depicted in Figure 1(b), where the order may be confirmed either by HTTP or by FTP.

5.3 Informational Aspect

Comparing the informational aspect of business and technical processes can be done by examining the factors that affect concepts of process information as well as their content and structure.

The design of business processes focus on the information need of the business parties, and the need to exchange information between business activities. The concepts in the process information will closely resemble the concepts used in the business (such as "customer", "offer" etc.), and also their content and structure.

The information content used when designing a technical process is the same as those used in a business process, since the process should support required information content in order to support the business. A difference between the business and technical processes is however the structure of the information. The structure simply needs to be adjusted to fit to the existing services and technologies. In addition to the change of structure, the protocols used might require the addition of extra service or protocol specific concepts, such as system or transaction identifiers.

Given the above discussion, we identify three different types of realization of the informational aspect:

- Extension/exclusion – it is possible that the technical process extends or the concepts identified in the business process, for example for handling technology dependent information such as transaction identifiers. Lack of systems supports for some business concepts will also entail the exclusion of concepts in the technical process.
- Projection – concepts defined in the business process might correspond to one or more concepts in the technical process. For instance the concept "product" in the business process may correspond to the concept "item" and "article" in the technical process as those concepts are used by underlying services.
- Redundancy – an information concept in the business process might be used more than once in the technical process. For example, on the technical level a single order might need to be duplicated in order to send it to both to the production and logistic systems.

5.4 Organizational Aspect

Business processes and technical processes utilises the concept of "roles" differently.

An early step when designing a business process is to define the roles that each business party has. Rights to processes and activities are later assigned to the roles. This means that the role concept in a business process is designed according to the business parties that are participating in the process.

In contrast to business processes, technical processes deal with resources and "parties" in form of services provided by systems.

Thus, when guiding activity invocation control, roles are assigned to system services.

Based on the outlined, the organizational aspect of a business process may be transformed in a corresponding technical process by the following realization type:

- Role Mapping – a role in the business process corresponds to several roles in the technical process, or several roles from the business process correspond to a single technical process role. For instance, the business process depicted in Figure 1, a single role, "order facilitator" might be responsible for both "confirm order" and "process order" activities. Invocations of those activities in the technical process might be assigned to different roles – "accepting order system" and "processing order system".

5.5 Transactional Aspect

Comparing two processes from the transactional perspective involves examining similarity in transactional behaviour reflected by specified transaction scopes and models.

The focus when designing a business process is to keep the process aligned with business contract rules. Thus, transaction design is based solely on business criteria. When an error occurs the process must return to a valid state by withdrawing results of completed activities. If by the contract, intermediate results are required to be visible, the process recovery is specified by compensations; otherwise, a valid state is obtained simply by cancelling what is done.

When designing technical processes it must be ensured that no systems are put in an inconsistent state with respect to the overall process. In case of errors the focus will be on service recovery, rather than on business contracts. The recovery methods (atomic vs. compensation) are specified according to business criteria, but also based on characteristics of services (data- vs. non-data based) as well as transaction support of back-end systems.

Thus, transactional aspect of a business process, with existence of supported systems and services, may be transformed in a technical process, by two realization types:

- Scope resize - the boundaries of a transaction differ in the business process and technical process. For instance, following a contract, in the business process it may be required to perform "customer registration", "order acquisition" and "order processing" activities within a transaction, while in the technical process, "customer registration" is not designed as part of the transaction, because underlying service is not transactional (as customer data need not to be deleted if the order is cancelled).
- Model alteration - the model (atomic vs. business) of a transaction differs in the business process and the technical process. As an example, in the business process, it may be required for a transaction, processing an order in several steps (i.e. activities), to provide intermediate results, and accordingly, to roll-back by compensations. However, in the technical process the transaction might be using a two-phase commit mechanism, as the order processing operations are data-based and compensating activities are not, therefore, designed.

As the realization types are defined, some dependencies among them may be observed. For instance, redundancy of an information concept in a technical process (such as processing of a same order by several systems) may be seen as the specialization in the functional aspect. As another example, conditional branching in a technical process may also origin from the specialization to available protocols/services. These examples illustrate that the realization types are related, i.e. that differences in the realization of one aspect may cause changes in another aspect.

Based on the identified differences in realizations of business processes and technical processes, in the following section, we discuss abilities for integration of those two specifications in a software system.

6. LIMITATIONS OF THE LAYERED ARCHITECTURE

As stated earlier, both business and technical processes have advantages. A realization that closely follows the business process will enable easy feedback to business actors in the form of the current process state. By implementing a technical process technical issues such as synchronization can be handled. Ideal would be to have an architecture with two realizations; one "pure" realization of the business processes unaffected by system issues, and one realization of the technical process that handles system issues. An architectural approach that enables this is layering [19].

In a layered approach, the business process is layered on top of the technical process. The executable business process reflects the states in the business, while technical issues are dealt with in the technical process. The dependency between the layers is this case is a "use" dependency [20], thus the executable business process uses the technical process to perform system actions.

The use of layers of executable processes is not new, it has been proposed before as a mean to provide graphical process interfaces to end-users [21]. Yang et al. [22] also describes an approach using "vertical" layers of processes, each layer pertaining to a certain business domain. However, in this chapter we will examine the special case where a layered approach can be applied to join a business and a technical process, without the use of vertical layers.

When executing the layers of processes, the business layer must be synchronized with the technical layer (see Figure 2). A basic criterion for this synchronization is that the technical process must be designed in such a way that it is a *lossless* realization of the business process. A lossless realization of a business process enables the same tracking of business goals as the original process. This means that it must be possible to perform tracking on all five aspects of the business process, even though the technical process is implemented taking system issues into consideration. For example, it must be possible to explain the current (business) process state to a business person even though it is realized as a technical process. As described earlier, the design of the layers adheres to different design principles, thus it might not be possible to use a layered approach in all cases. In the following sections we examine which types of business process realizations that hinder a lossless realization, and thus hinder the use of the layered approach. The examination is based on the previously described realization types.

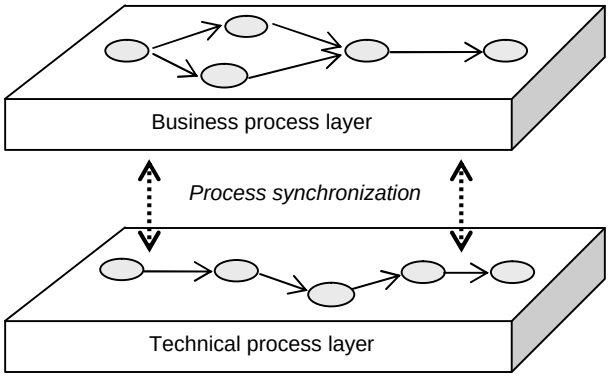

Figure 2. Overview of the layered approach.

Realization of the functional aspect of a business processes can affect its set of activities and their pre and post conditions on the business level. When using a layered architecture, the realization of the business process must adhere to the following design requirements:

- Named activities in the technical process must be *aggregated*/mapped into a single activity name in the business process, otherwise it is not possible to determine which activity that are executing on the business level. Thus, two named activities in the business process cannot correspond to a single activity in the technical process.
- *Conditions* in the technical process must be designed such that the activity pre-conditions are the same or weaker in the technical process. Post-conditions in the technical process must be the same or stronger. Note that both set of conditions will be checked at runtime, since both the business process and the technical process will be executed, each in their own layer. Note also that this design requirement is the same as when designing inheritance hierarchies in object-oriented systems [23].

Note that s*pecialization* of activities in the business process can occur without affecting the synchronization of the processes.

Realization of the behavioural aspect, including sequencing, parallelism and conditional statements can also affect the possibility to use the layered design. With regard to the behavioural aspect the realization must adhere to the following:

- The *concurrency* in form of parallel flows and synchronizations must be designed such that parallel flows are avoided in the technical process, where sequential flow is used in the business process. Without following this requirement, the state of the control flow on the business level cannot be determined.
- The *sequencing* of the activities in the business processes must be reflected in the ordering of the corresponding activities in the technical process. In essence this ensures that the control flow of the activities in the technical process can be depicted by the control flow of the business process.
- *Conditional* control flow must be designed such that each (optional) path in the business process corresponds to at least one unique path in the technical process. If this is not the case it is not possible to determine which path in the business process that is being executed.

The informational aspects concerns realization of the technical process in the form of projection and extension of the concepts present the business process. In addition to this, redundancy on the concept instance level might be introduced. Keeping the technical and business process synchronized with regard to the informational aspect entails following the following design requirements:

- *Exclusion* of concepts in the technical process may not be introduced. However, extensions to handle technical issues might be introduced.
- *Projection* of concepts in the business process must be done such that each concept is represented in the technical process. If the layers contain differences in data naming, a mapping is necessary.

Note that *redundancy* may be introduced in the form of handling duplicated information at the technical level, without affecting the synchronization of the processes.

When realizing the organizational aspects, roles in the business process must be mapped to systems in the technical process. To keep a lossless realization this mapping must adhere to the following:

- Parties in the business process that is responsible for executing the business activities must own, or be responsible for, the realizations of these activities in the technical process. For example if a party is responsible for an activity on the business level, and that activity is performed by a system owned by another party, it is unclear who got the actual responsibility for the correct outcome of the activity.

The transactional aspect concerns the realization of transaction models and the design of transactions boundaries. In order to provide the same transactional integrity as the business process the realization must:

- The *transaction model* of the activities in the in the technical process must support the same, or a higher level of transaction model compared to the business process. For example, if the business process is designed using long running transactions, the technical process must support atomic transactions or long running transactions.
- The *transaction boundaries* must be realized such that the boundaries in the technical process do not overlap those defined in the business process.

As presented above, restrictions on the use of most realization types must be introduced to yield a lossless realization. To achieve a lossless realization of all aspects of a business process is therefore difficult. For example, the real-world process in Figure 1 is not a lossless realization. In this case, a comparison of the behavioural aspect of the business and technical process reveals a use of the sequencing realization type that is not lossless. Even in those cases a total “losslessnes” can not be achieved, awareness of the realization types can lead to a process design that is closer to the original business process.

7. CONCLUSION

In this paper we have addressed the realization gap between business and technical processes. On an abstract level this gap can be described as a business versus information technology gap. However, our contribution lies on a more concrete level as we, based on a process description framework, define a set of realization types that describe the possible transformation needed when constructing a technical process from a business process. In addition, we have presented design requirements that need to be followed in order to create a lossless realization of a business process.

This work has been conducted in several steps. It started with the process case presented in Figure 1. When examining this process, it was evident that the expressiveness of the process notation was used to cover both business and technical needs. This motivated us to make a structured examination of how technical considerations affect realization of a business process. The second step of the work was to examine under which conditions a business process and a technical process could be integrated into a single architecture. The first part of the work resulted in the definition of business and technical processes presented in Section 2, as well as the realization types presented in Section 5. The second part of the work resulted in the notion of lossless realization, and the design requirements for lossless realization presented in Section 6.

The presented result can be applied in several ways. Firstly, the definition of business and technical process along with the presented realization types can be used as a conceptual framework for discussing executable process design in the context of existing systems. Secondly, the presented guidelines on lossless realization can be used to govern the design of software architectures that enable a closer integration between the business and technological view of processes.

The presented realization types are a first step in creating a model-driven tool that supports realization of business processes. Further work with the aim to create such a tool entails examining the relations between the realization types, as well as a closer examination of the parameters of the realization function, R(S,B). Additional examination of the realizations types includes their formalization, and describing dependencies among them. Some dependencies among realization types are briefly mentioned in the end of Section 5. However, these dependencies and the order in which to apply realization types need to be further examined.

Considering future work with the realization function (R), we defined it as having two parameters, the business process and system constraints, R(S,B). However, in this paper we have not examined the relative importance of system constraints (S) and the business process (B). Neither have we categorized the system constraints. As mentioned in the introduction, there might be cases where the power of executable process languages is used to solve mainly technical issues. In these cases technical issues (S) in the realization process might actually have a bigger influence on the final outcome than business aspects (B). Future categorization of system constraints might enable relating the realization types with typical system constraints, thereby forming the basis for tool based business-to-technical process transformations.

8. ACKNOWLEDGMENTS

The authors would like to thank Sandvik for providing the input to the example case presented in Section 2. This work is a part of the Serviam project, partly funded by the Swedish Agency for Innovation Systems.

9. REFERENCES

[1] Papazoglou, M., Georgopoulos D. Special issue on service oriented computing. Communications of the ACM, 10, 46 (Jan. 2003), 24-28.

[2] Piccinelli G., Zirpins, C., Lamersdorf W. The FRESCO Framework: An Overview. Proceedings of the 2003 Symposium on Applications and the Internet Workshops. IEEE Computer Society (2003), 120-126.

[3] Sharp, A., McDermott, P. Workflow Modeling. Artech House, Inc., Boston, USA, 2001.

[4] Hohpe, G. et al. Enterprise Integration Patterns, Addison. Wesley, Oct 2003.

[5] Bubenko, J.A., Jr., Wangler, B. Objective driven capture of business rules and of information systems requirements. In Proceedings of the IEEE Systems Man and Cybernetics '93 Conference (Le Touquet, France, October 1993), 670-677.

[6] Grover, V., Fiedler, K., Teng, J.T.C. IEEE Transactions on Engineering Management, 41, 3 (Aug. 1994), 276 – 284.

[7] Yu, E. S. K., Du Bois, P., Dubois, E., Mylopoulos, J. From Organization Models to System Requirements: A 'Cooperating Agents' Approach. In Proceedings of the Third International Conference on Cooperative Information Systems (CoopIS'95) (Vienna, Austria, May 1995), 194-204.

[8] Rolland, C.,Prakash, N. Bridging the Gap Between Organisational Needs and ERP Functionality. Journal of Requirements Enigneeering, 5, 3, 2000, 180-193.

[9] Rolland, C.,Prakash, N. Matching ERP System Functionality to Customer Requirements. 5th IEEE International Symposium on Requirements Engineering (RE 2001) (Toronto, Canada, Aug. 2001). IEEE Computer Society (2001), 66-75.

[10] Kleppe, A., Warmer, J., Bast, W., MDA Explained, Addison-Wesley, Apr 2003.

[11] Weis, T., Ulbrich, A., Geihs, K. Model Metamorphosis. IEEE Software (sept./oct. 2003), 46-51.

[12] BEA, IBM, Microsoft, SAP and Siebel. Business Process Execution Language for Web Services (BPEL). http://www-106.ibm.com/developerworks/library/ws-bpel/, June 9 2004.

[13] Aalst van der, W., Hofstede, A., Aldred, L. Yet Another Workflow Language (YAWL). http://www.citi.qut.edu.au/yawl/index.jsp, June 9 2004.

[14] White, S. Business Process Modeling Notation Version 1.0, The Business Management Initiative, May 2004.

[15] Jablonski, S. A Software Architecture for Workflow Management Systems. In Proceedings of the Ninth International Workshop on Database and Expert Systems Applications (DEXA'98) (Vienna, Austria, August 1998). . IEEE Computer Society, 1998, 739-744.

[16] Rausch-Scott, S. TriGSflow – Workflow Management Based on Active Object-Oriented Database Systems and Extended Transaction Mechanisms. PhD Thesis, University at Linz, 1997.

[17] Bernstein, P., Hadzilacos, V., Goodman, N. Concurrency Control and Recovery in Database Systems. Addison-Wesley, 1987.

[18] Garcia-Molina, H. Modeling Long-Running Activities as Nested Sagas. IEEE Data Engineering Bulletin, 14, 1, 1991, 14–18.

[19] Bass, L., Clements, P. and Kazman, P. Software architecture in practice. Addison Wesley, 1998.

[20] Parnas, D. Designing software for ease of extension and contraction. IEEE Transactions on Software Engineering, March 1979, 128-138.

[21] Johannesson P., and Perjons, E. Design principles for process modelling in enterprise application integration, Information Systems, 26, 2001, 165-184.

[22] Yang, J., Papazoglou, M. Interoperation Support for Electronic Commerce. Communications of the ACM 6, 43, 2000, 39-47.

[23] Meyer, B. Applying Design by Contract. IEEE Computer, 25, 10, (Oct. 1992), 40 – 51.