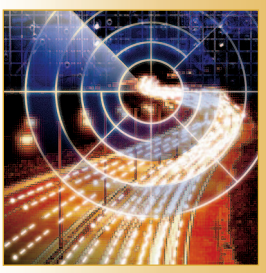
Editors: Michael Huhns • *huhns@sc.edu*
Munindar P. Singh • *singh@ncsu.edu*



Moving from Business to Technology with Service-Based Processes

In most organizations, business process realizations must be aligned with existing systems, which can impose specific requirements. Currently, however, there is no way to track the alignment between business processes and corresponding technical implementations. The authors' proposed framework offers a systematic way to classify and assess technical realizations of business processes.

Jelena Zdravkovic
University of Gävle and Royal Institute of Technology, Sweden

Martin Henkel and Paul Johannesson
Stockholm University and Royal Institute of Technology, Sweden

Software services such as e-procurement and e-payment are fundamental to large-scale systems that cross enterprise boundaries to form e-business marketplaces. To enable complex business collaborations, executable processes can combine these services into service networks[1] in which executable process descriptions coordinate service interactions. (We can write such descriptions with the Business Process Execution Language for Web Services; BPEL, www-106.ibm.com/developerworks/library/ws-bpel.)

The use of process technologies to model and implement service interactions has several advantages over application-centric solutions. From a business perspective, software services more closely match the services a company offers, given that *business processes* are modeled to follow business activities, events, and message exchanges. From a technical perspective, software services offer the ability to structure and manage system-to-system communications by designing *technical processes* to fit in with legacy systems and previously implemented services.

A process designer must consider both of these perspectives. Rather than imposing limitations on the final design, the technical process should ideally be designed to directly correspond to the business process. Constraints in legacy systems often lead to differences, or *misfits*, between desired business processes and implemented technical processes, so that the resulting system fails to support all aspects of the business.

In this article, we examine crucial criteria for constructing technical processes that support business processes. The research community has studied the general gap between enterprise business processes and information system functionality,[2,3] but our work differs in two aspects. First, we examine misfits, focusing mainly on executable process specifications. Moreover, we attempt to

 1089-7801/05/$20.00 © 2005 IEEE Published by the IEEE Computer Society

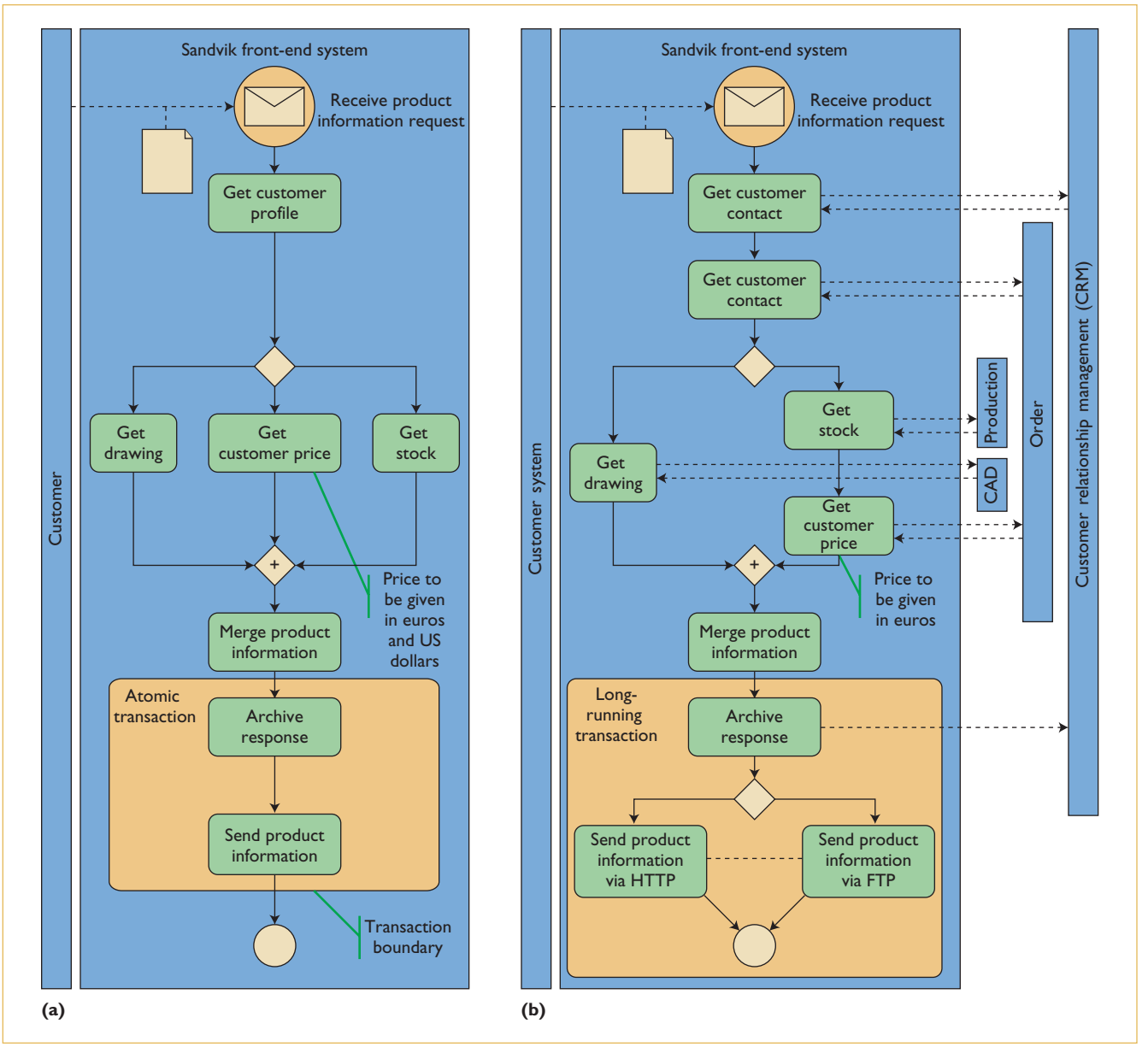


Figure 1. Sandvik's front-end process model. (a) The business process differs from (b) the realized technical process in the number of activities and their ordering, exchanged messages, and transactions. (CAD stands for computer-aided design.)

bridge the gap between existing business- and technology-dependent processes by introducing the notion of *realization extents*, which discern the fit between those processes into four levels of quality.

Realizations of Business Processes

Figure 1 shows a business process realized as a technical process. This example process is based on cases provided by Sandvik, a global company for industrial materials engineering. Sandvik is our partner on a project that aims to define patterns for building processes based on Web services. One of the company's key concerns is how to integrate existing enterprise resource planning (ERP) systems into new software services, and then combine and coordinate individual services into more complex interactions.

Figure 1a depicts the company's basic business process for supplying customers with product information. The model is expressed in Business Process Modeling Notation (BPMN; www.bpmi.org[4]), which visually models process management, but it can be converted to a process language, such

as BPEL or Business Process Modeling Language (BPML; www.bpmi.org[5]), or to a collaboration specification (see www.ebxml.org or www.rosettanet.org). After establishing mappings between BPMN semantics on one side and platform-level languages and specifications on the other, we can automate the conversion.

Upon receiving a product information request, the first step in the business process is to gather information about the customer and create a customer profile, which includes order history and contact information. Three activities then execute in parallel to fetch a product drawing, get current stock levels, and calculate the customer's price for the product. After the necessary product information is collected and merged, it is archived and sent to the customer. Because the archived copy shouldn't be available if the product information hasn't reached the customer yet, archive and send activities are bound in an atomic transaction (AT).

Figure 1b shows the technical process, which is based on existing ERP services, depicted as customer relationship management (CRM), order, production, and computer-aided design (CAD). These systems impose several technical constraints that affect the business process:

- The process obtains a customer profile using two activities instead of one because one activity gathers customer contact information from the CRM system and another fetches the order histories from the order system.
- To calculate product price, the order system requires the stock's location (to get transport cost and calculate currency), which means the process can't obtain the customer's price concurrently with the stock information.
- The customer can receive product information either as an HTTP message or in an FTP file (the company supports both transport protocols).
- The CRM system doesn't support two-phase commit transactions, so the process must explicitly remove archived responses to product requests if the product information doesn't reach the customer (for example, the customer is unavailable for some reason). The process thus defines the transaction as long-running.

As Figure 1 shows, the company must perform several activities in the form of a process to provide customers with information about company products; each process instance handles a customer request as a separate *case*, which is uniquely identified by customer data, such as name and required products.

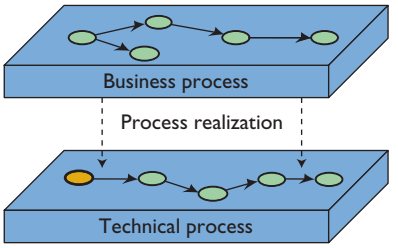


Figure 2. Alignment of business and technical processes. To realize a business process, a technical process must track the original process state.

A case begins when the process receives a customer request. This request passes through several states — from the point at which the process retrieves the customer profile, to when it obtains price stock and drawings, and finally, to when the process sends the product information to the customer. Activities execute differently in response to specific cases. In Figure 1a, for example, one customer might need the product price calculated in euros, whereas someone else might need US dollars. To satisfy both customers, the process must support both currencies.

In general, business processes are designed to support a group of *required cases*. To keep technical and business processes aligned during case execution, the technical processes must track the original processes' states for all required cases. This ensures that, from the business process view, all business states are realized (see Figure 2).

In Figure 1a, for example, the business model obtains the price concurrently with the product stock, whereas the technical model processes these activities sequentially in Figure 1b. For those customers who expect to get the product price only if the stock information is available (sequentially), the technical process realizes the business process; for other customers, it doesn't. Furthermore, using HTTP and FTP to send product information might be perfect for some customers — in those cases, the technical process tracks the business states — but for those who can't accept either protocol, the technical processes don't track the business states. Ultimately, creating a technical process that strictly realizes a given business process is challenging. Process designers must be able to discern whether a business process can be realized "ideally," in some limited way, or not at all.

To clarify this question, we delineate four *real-*

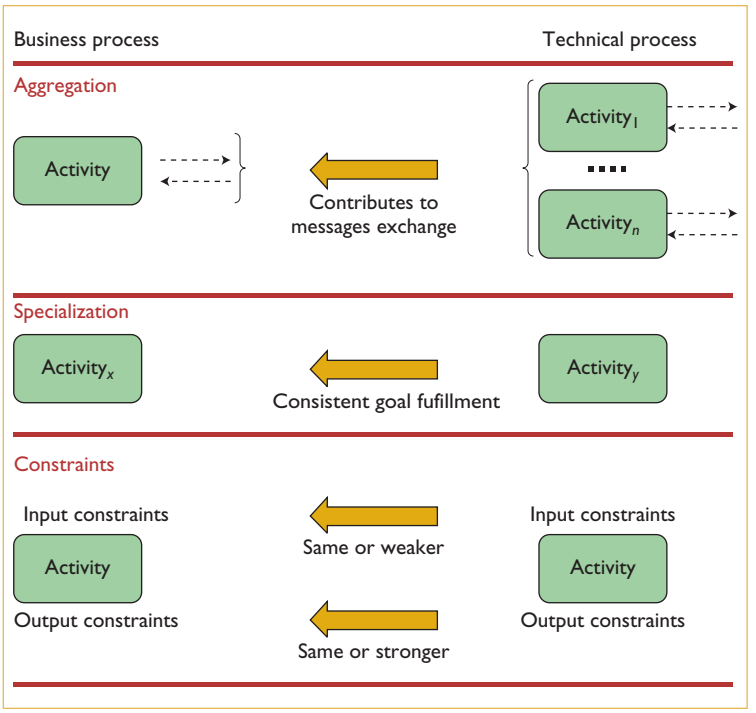


Figure 3. The functional aspect. Applying three rules (symbolized by arrows) during process design enforces a lossless process realization.

ization levels to describe a technical process's ability to track a business process's state:

- *Lossy*. The technical process doesn't track business process states for any required case.
- *Constrained*. The technical process doesn't track business process states for all required cases.
- *Lossless*. The technical process tracks business process states for all required cases.
- *Exceeded*. The technical process tracks original business process states for required and additional (that is, not required) cases.

The importance of distinguishing between lossless and lossy realizations is clear: they separate "ideal" from "infeasible" business process realizations. In many circumstances, however, existing services enable the realization of a business process for a certain group of cases (a constrained realization). Detecting the situations in which existing services provide more capabilities than required is also important because they indicate that more resources are engaged than needed.

Rules for Technical Process Design

In our previous work,[6] we used a framework that contained five main aspects of process design: functional, behavioral, organizational, informational, and transactional.[7,8] We must consider all five when designing a technical process.

Functional Aspect

Three elements describe an activity's functionality: the activity name describes the goal to be fulfilled, input and output messages define information exchange, and input and output constraints describe pre- and postconditions.[7,8] In a business process, business rules govern activity functionality. If a business requires that a product's price be determined from customer information, for example, it leads to the activity "get customer price," for which the input message includes the product cost and customer's order history; the output message is a customized price. A constraint on this activity's output might be to calculate the price in the customer's local currency.

When a business process is realized, the technical process might also require functionality changes: activities in the technical process might aggregate exchanged messages differently than activities in the business process, for example, or they might impose different constraints. Such changes could hamper the technical process's ability to represent business states, thus leading it to become a lossy realization of the business process.

Alternatively, a business process becomes lossless if the technical process design fulfills the rules illustrated in Figure 3:

- *Aggregation*. Activities should be aggregated such that one activity's message exchange contributes to the message exchange (and thus, the goal fulfillment) of a single activity in the business process. If the activities in the technical process adhere to this rule, we can relate a single activity, or group of activities, to a single activity in the business process. This ensures that the business can monitor the process's state on a per-activity basis. In Figure 1, for example, the business process retrieves the customer profile as a single message (activity), whereas the technical process requires two activities to provide this functionality because different ERP systems hold the contact and order history information.
- *Specialization*. Business process activities can be specialized in the technical process, but only so that the resulting technical process activities fulfill the business process activities' goals. Specialization means adding details (such as the selection of communications protocols) to

accommodate the activity in a technical system environment. The specialization rule ensures that the process designer doesn't introduce conflicting specializations. In Figure 1, for example, the business activity "send product information" is specialized to use both FTP and HTTP protocols. Here, a conflict would be evident if the customer's receiving activity didn't support either protocol. In this case, the customer doesn't have the capability to receive product information, and thus can't participate in the realized business process.

- *Constraints*. Constraints between the processes must be mapped such that the technical process activities' input constraints are the same or weaker than in the business process; output constraints must be the same or stronger.[9] A condition can be made stronger in the technical process, for example, by limiting the range of valid input values. If the activity "get customer price" in the business process is defined to handle euros as valid output, for example, the corresponding activity in the technical process must also support the currency.

Violating these rules leads to lossy business process realizations. Existing services can sometimes fulfill these rules as well as provide nonrequired functionality. The resulting business process will be realized at the exceeded level if the technical process can support alternative activity aggregations, weaker input constraints, or alternative specializations. Alternative realizations of activities and the potential to handle a wider range of process cases (weaker input constraints) help the process realization accommodate future changes in both technology and business processes.

Depending on case data, the technical process's specialization, aggregation, and constraints might support certain cases, but not others. Some customers might require protocols other than HTTP or FTP to receive product information, for example. Designing the functional aspect of the technical process such that it doesn't support all process cases will lead to a constrained realization of the business process.

Behavioral Aspect

A process's behavioral aspect describes when an activity is executed in relation to other activities. Three basic control-flow constructs express ordering: sequence, parallel execution, and conditional branching. In a business process, business rules govern the use of these constructs — the activity "receive payment" should be completed before the activity "ship product" commences, for example. However, when realizing a business process as a technical process, existing services can limit their use: dependencies between services could prohibit them from being executed in parallel, as the earlier example illustrated.

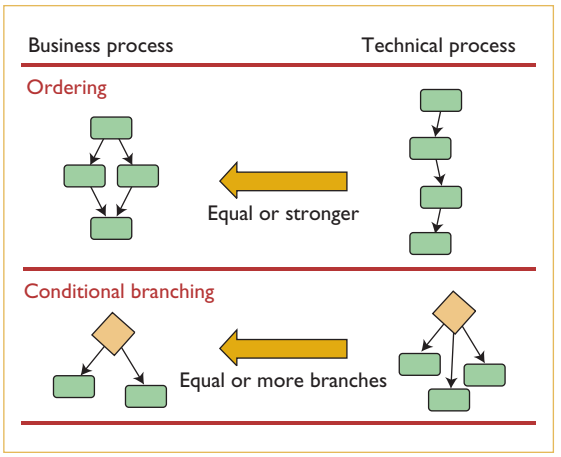


Figure 4. Behavioral aspect rules. A lossless realization is ensured if the technical process provides a stronger activity ordering and additional conditional branches compared to the business process.

To be lossless, the technical process must follow the rules illustrated in Figure 4:

- *Ordering*. In a technical process, activity ordering must be the same as, or stronger than, in the business process. This means that parallel flows in a business process could be realized by using the same (parallel) order or by using the stronger sequential ordering in the technical process. In contrast, a sequence in a business process must be realized in the same way in a technical process to ensure that the technical process control flow mirrors the control flow of business process activities.
- *Conditional branching*. Control flow must be designed such that every branch in the business process corresponds to at least one branch in the technical process. If this isn't the case, determining the executed branch in the business process is impossible.

Breaking these requirements means that the business process is unrealizable in its current form. However, if existing services support the ability to

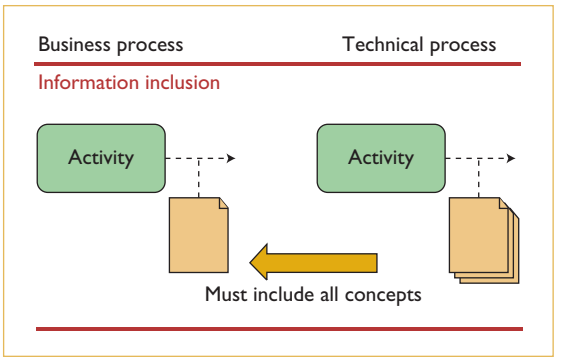


Figure 5. Informational aspect rule. All business concepts must be represented in the technical process, so messages sent from a technical process activity can contain the same, or more, concepts compared to the business process.

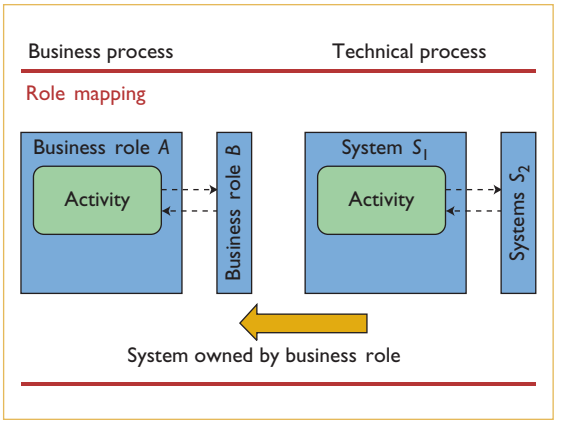


Figure 6. Organizational aspect rule. A lossless realization is ensured by the fact that role A *owns system* S_1*, whereas role* B *owns system* S_2.

arrange activities in both parallel and sequential flow, or if the existing services support additional conditional branches, an exceeded realization of the business process is possible. Even though the technical process can track all the business process's states, it might have limited support for case data; this creates a constrained realization of the business process. Two different branches of a process might handle customer credit ratings, for example — one for credit level A and one for credit level B. The process might not be able to handle some of the level-B customers because the level-B conditional branch simply isn't supported by existing services. In this situation, actual case data decides whether the technical process can realize the business process in a lossless way. In this case, the realization is constrained because the technical process can't handle all case data.

Informational Aspect

A process's informational aspect involves its internal data and the data that it exchanges with its external environment. In a business process, these concepts are modeled to resemble customers, orders, and other documents exchanged with business partners. When the business process is realized as a technical process, well-defined information structures represent business concepts, but the process designer might need to add further concepts (to represent technical concerns such as transaction and system identifiers, for example).

To losslessly realize a business process, the designer must follow the rule of *inclusion* — that is, all business concepts must be included in the technical process. As Figure 5 shows, this simple requirement means that the business should always be able to trace information concepts in a business process to their equivalent concepts in the technical process.

A technical process that doesn't include all of the business process's information might still be able to handle some of the required cases at a constrained level (working for customers with a single delivery address, for example, but not for those with multiple addresses).

Organizational Aspect

The organizational aspect describes the responsibility for activity execution. This responsibility is typically assigned to business roles, such as subsupplier and customer, which are later responsible for executing various process activities. By using roles in this manner, the process designer can dedicate and control the responsibilities of the parties engaged in the process. In a technical process, for example, a business partner responsible for receiving and handling a business document might be represented by a service that fulfills the partner's obligations. A third party might host the service, but the partner defined in the business process is still responsible for ensuring the correct outcome.

The process designer must follow the *role-mapping* rule to ensure that the execution of technical process activities corresponds to the responsibilities defined for the various business roles in the business process. As Figure 6 shows, technical realizations must be owned by or under the supervision of the parties who are responsible for the corresponding business activities.

Breaking the role-mapping rule results in lossy realizations of business processes because it isn't clear who is actually responsible for executing a

service that isn't owned by the partner defined in the business process. As a result, no one can be held accountable if a failure occurs.

As with the other aspects, existing services can support more than what the business process requires in the organizational aspect — particularly if several business parties support the same (software) service. If the responsibility for executing a business activity shifts to another party, the process designer can simply reconfigure the technical process to use the new party's service. This kind of exceeded realization provides businesses with flexibility in distributing their responsibilities.

Usually, the process designer assigns business roles at design time, but in a technical process, the executing process can utilize case data to discover and use services at runtime. Messages might be routed to a supplier that currently has a product in stock, for example. If all of the supplier's services are appropriately mapped to business roles, process realization remains lossless for all case data. If clear responsibility statements don't back up the technical process, the executing process might use "uncontracted" (and thus, unreliable) suppliers, which can result in a constrained realization of the organizational aspect by failing to properly support all required cases.

Transactional Aspect

The transactional aspect governs the consistent execution of a set of activities. Because loosely coupled process activities can be short or long in duration, process transactions must comply with two different models:

- The *atomic transaction* (AT) model[10] controls a set of shorter activities such that the outcome is visible only when all activities within the transaction finish successfully. In case of errors, systems typically use low-level, two-phase commit protocols to automatically roll back these activities. (If the activities are implemented in the form of Web services, then the WS-AtomicTransaction specification [www-128.ibm.com/developerworks/library/specification/ws-tx] coordinates the services in the AT model.)
- The *long-running transaction* (LRT) model rules more durable activities:[11] each activity's outcome is globally visible as soon as the activity is completed, regardless of the outcome of other activities. In case of errors, the transaction is rolled back by compensating the activities that have successfully completed. The WS-Business-Activity specification (www-128.ibm.com/developerworks/library/specification/ws-tx) fully implements the LRT model's concepts in governing the coordination of long-durable Web services.

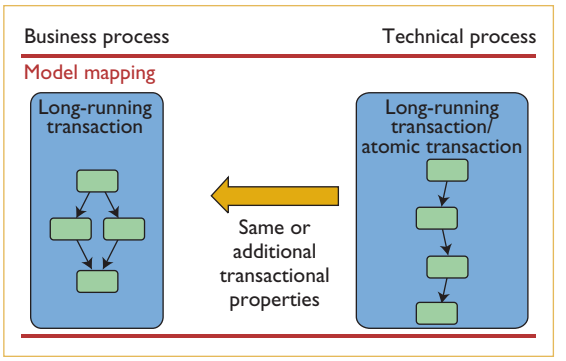


Figure 7. Transactional aspect rule. The technical process must support at least the transactional model defined in the business process.

Depending on existing services' capabilities, a technical process can enforce the use of a transactional model other than what the business process requires. For example, order-processing activities in a business process might be long running to let a company's purchasers control intermediate documents, whereas order-processing services in the technical process might use atomic transactions (short-running activities). This means that the compensating activities the business requires aren't present in the technical process.

Activities in the technical process must have the same transactional properties as activities in the business process. If an activity from a business process commits globally (is long running) upon completion, the technical process must be realized identically. Therefore, lossless realization of the transactional aspect requires *model mapping* — that is, activities in the technical process must support at least those transactional properties defined for activities in the business process (see Figure 7).

If the model-mapping rule can't be applied (from the business perspective, this means the models differ), the realization is lossy. In Figure 1a, for example, the business process requires that product information not be archived until the customer successfully receives the information, which means that the activities should use the AT. In the technical process, however, the transaction model is long running: the activities for sending FTP or HTTP messages can't be atomic due to the use of nontransactional message protocols. In contrast, if

the existing services can support both atomic and long-running models — if resources for both two-phase commit and compensations are available — then the technical process can provide an exceeded realization.

Realization Assessment

We can use this set of rules to examine a technical process on a per-aspect basis and discern how well it realizes the corresponding business process. For realizations that aren't lossless, we use case data to further examine whether they are constrained for each aspect. This analysis yields a list of required modifications to the technical process to support the business process.

To illustrate, we can use the example from Figure 1:

- By examining the functional aspect, we see that each activity in the business process is represented by one or more activities in the technical process; thus, the aggregation rule holds. The specialization rule is also satisfied because the HTTP and FTP protocols fulfill customers' requirements. In contrast, the constraint rule isn't satisfied in the technical process because the product information requester can obtain product prices only in euros (the business process also requires dollars). Yet, because Sandvik's customers primarily use euros, the mapping of the constraint is satisfied for these cases. On balance, the functional aspect is constrained.
- For the behavioral aspect, the technical process uses the sequential flow for the "get stock" and "get customer price" activities instead of the parallel flow in the business process. Furthermore, a conditional branching has separate activities for FTP and HTTP. However, neither of these modifications breaks the stated rules of ordering or conditional branching.
- To determine the realization level for the informational aspect, we must look beyond BPMN models in Figure 1 (because the notation does not visualize information modeling) and compare the message documents in both processes. By investigating the inclusion of required information concepts in the technical process, however, we conclude that the rule holds.
- The organizational aspect requires information about each activity's owner, which Figure 1 doesn't show. However, Sandvik is responsible for executing all activities except those within the customer system, which means that a clear role mapping exists between the business roles and the systems that execute the process.
- Examining the transactional aspect, we see that the activities in the technical process don't have the AT properties the business process requires; thus, the realization fails the model-mapping rule.

If Sandvik could redesign its existing services to support the AT model for required transactions, this lossy realization could be improved to merely the constrained level. By updating the service for product price calculation to provide results in both euros and dollars, the company could even create a lossless realization of the business process.

Process designers can perform this type of assessment in the design stage to create a plan for lossless realization. Later, when doing maintenance on the process because of changing business requirements, the designer can apply the realization rules to perform an impact analysis.

For our future work, we see a need to identify and classify system constraints to enable a straightforward way of handling technical process design rules. A process design environment that considers both system constraint documentation and the ability to trace process realizations would be a powerful tool for combined system and business integration.

Acknowledgments

We thank Sandvik for providing the input to the example case.

References

1. G. Piccinelli, C. Zirpins, and W. Lamersdorf, "The FRESCO Framework: An Overview," *Proc. 2003 Symp. Applications and the Internet*, IEEE CS Press, 2003, pp. 120–126.
2. J.A. Bubenko Jr. and B. Wangler, "Objective Driven Capture of Business Rules and of Information Systems Requirements," *Proc. IEEE Systems Man and Cybernetics Conf.*, IEEE Press, 1993, pp. 670–677.
3. C. Rolland and N. Prakash, "Bridging the Gap between Organizational Needs and ERP Functionality," *J. Requirements Eng.*, vol. 5, no. 3, 2000, pp. 180–193.
4. S. White, *Business Process Modeling Notation, version 1.0*, Business Management Initiative, May 2004; www.bpmi.org.
5. A. Arkin, *Business Process Modeling Language, version 1.0*, Business Management Initiative, Nov. 2002, www.bpmi.org.

6. M. Henkel, J. Zdravkovic, and P. Johannesson, "Service-Based Processes: Design for Business and Technology," *Proc. 2nd Int'l Conf. Service Computing*, ACM Press, 2004, pp. 21–29
7. S. Jablonski, "A Software Architecture for Workflow Management Systems," *Proc. 9th Int'l Workshop Database and Expert Systems Applications*, IEEE CS Press, 1998, pp. 739–744.
8. S. Rausch-Scott, "TriGSflow: Workflow Management Based on Active Object-Oriented Database Systems and Extended Transaction Mechanisms," PhD thesis, Dept. of Information Systems, Univ. at Linz, 1997.
9. B. Meyer, "Applying Design by Contract," *Computer*, vol. 25, no. 10, 1992, pp. 40–51.
10. P. Bernstein, V. Hadzilacos, and N. Goodman, *Concurrency Control and Recovery in Database Systems*, Addison-Wesley, 1987.
11. H. Garcia-Molina, "Modeling Long-Running Activities as Sagas," *IEEE Data Eng. Bull.*, vol. 14, no. 1, 1991, pp. 14–18.

Jelena Zdravkovic is a PhD candidate at the Royal Institute of Technology in Stockholm. Her technical interests include process modeling, alignment between business and IT, and service-oriented architectures. Zdravkovic has an MSc in computer and system sciences from the Belgrade University of Electrical and Computer Engineering and an MBA in e-commerce from the University of Gävle. Contact her at jzc@dsv.su.se.

Martin Henkel is a PhD candidate at Stockholm University and the Royal Institute of Technology. His technical interests include service-oriented architectures and process integration. Henkel has an MSc in computer and systems sciences from Stockholm University. Contact him at martinh@dsv.su.se.

Paul Johannesson is a professor at the Royal Institute of Technology. His technical interests include the use of linguistic instruments in information systems, analysis patterns in systems design, process integration, and e-commerce systems design. Johannesson has a BSc in mathematics and a PhD in computer science from Stockholm University. Contact him at pajo@dsv.su.se.