Flexible Support of Healthcare Processes

Manfred Reichert and Rüdiger Pryss

Institute of Databases and Information Systems, Ulm University, Germany
manfred.reichert@uni.ulm.de; ruediger.pryss@uni-ulm.de

Abstract. Traditionally, healthcare information systems have focused on the support of predictable and repetitive clinical processes. Even though the latter can be often prespecified in formal process models, process flexibility in terms of dynamic adaptability is indispensable to cope with exceptions and unforeseen situations. Flexibility is further required to accommodate the need for evolving healthcare processes and to properly support healthcare process variability. In addition, process-aware information systems are increasingly used to support less structured healthcare processes (i.e., patient treatment processes), which can be characterized as knowledge-intensive. Healthcare processes of this category are neither fully predictable nor repetitive and, therefore, they cannot be fully prespecified at design time. The partial unpredictability of these processes, in turn, demands a certain amount of looseness. This chapter deals with the characteristic flexibility needs of both prespecified and loosely specified healthcare processes. In addition, it presents fundamental flexibility features required to address these flexibility needs as well as to accommodate them in healthcare practice.

1 Introduction

Traditionally, *process-aware information systems (PAIS)* have focused on the support of predictable and repetitive business processes, which can be fully described prior to their execution in terms of formal process models [46]. Characteristic examples of healthcare processes falling in this category include organizational procedures in hospitals, like medical order entry and result reporting, as well as administrative processes. In spite of several success stories on the uptake of process-aware information systems in healthcare and the growing process-orientation in this domain, *Business Process Management (BPM)* technologies have not been widely adopted in healthcare yet [17, 33].

A major reason for the low use of BPM systems in healthcare has been the rigidity enforced by them, which inhibits the ability of a hospital to respond to process changes and exceptional situations in an agile way [25]. When efforts are taken to improve and automate the flow of healthcare processes, however, it is of utmost importance not to restrict medical staff [5]. First attempts to change the function- and data-centric views on patient treatment processes failed whenever rigidity came with them. Variations in the course of a disease or treatment process are inherent to medicine, and to some degree the unforeseen event constitutes a "normal" phenomenon [22]. Hence, a sufficient degree of flexibility

is needed to support dynamic process adaptations in case of such unforeseen situations. Moreover, *PAIS flexibility* is required to accommodate the need for evolving healthcare processes [30], e.g., to integrate new medical devices, implement new laws, or change clinical guidelines (due to new empirical evidence). Finally, support for healthcare process variability is needed [29, 2]. For example, in a particular hospital, different variants of the oder entry process may exist whose concrete behavior and structure depends on various contextual factors like the status of the patient, the kind of medical examination ordered, or the concrete provider of the medical service [18].

For several years, BPM technologies have been increasingly used to support less structured business processes as well [24]. The latter include patient treatment processes and are often characterized as knowledge-intensive. Processes of this category feature non-repeatability, i.e., the models of two process instances (e.g., coordinating the treatment of two different patients) do not fully resemble one another. Generally, *knowledge-intensive processes* tend to be *unpredictable* as their exact course of action depends on situation-specic parameters [19, 21]. Usually, the values of the latter are unknown a priori and may change during process execution. Moreover, knowledge-intensive processes can be characterized as *emergent*, i.e., knowledge and information gathered during the execution of the process determines its future course of action. Consequently, respective processes cannot be prescribed at a ne-grained level at design time. In addition to *variability*, *adaptation*, and *evolution*, which are also needed in the context of predictable processes, they require *looseness*.

The vast majority of healthcare processes can be characterized by a combination of predictable and unpredictable elements falling in between the two extremes described above. While procedures for handling single medical orders or examinations are relatively predictable, complex patient treatment processes are rather unpredictable and unfold during process execution [17].

This chapter elaborates on advanced BPM concepts enabling process flexibility at the operational level. Emphasis is put on key features enabling process variability, process adaptation, process evolution, and process looseness. Based on them process-aware healthcare information systems, being able to flexibly cope with real-world exceptions, uncertainty and change, can be realized. Section 2 presents the conditions under which a process-aware healthcare information system needs to operate and illustrates the need for flexible healthcare process support in this context. Section 3 then discusses and structures the flexibility needs of both perspecified and loosely specified healthcare processes in detail. Sections 4 – 7 present concepts and techniques for properly addressing these flexibility needs. Section 8 deals with other approaches fostering process flexibility, whereas Section 9 concludes and summarizes the chapter.

2 Healthcare Process Characteristics

In the following, an impression of the characteristic properties of hospital working environments is provided to give an idea under which conditions process-aware

healthcare information systems need to operate. On one hand, this real-life description confirms the high need for process coordination in healthcare, on the other it emphasizes the non-suitability of rigid approaches when it comes to the automation of healthcare processes.

In a hospital, the work of clinical staff is burdened by numerous organizational as well as medical tasks. Medical procedures must be planned, ordered and prepared, appointments be made, and results be obtained and evaluated. Usually, in the diagnostic and treatment process of a particular patient various, organizationally more or less autonomous units are involved. For a patient treated in a department of internal medicine, for example, medical tests and procedures at the laboratory and the radiology department might be required. In addition, samples or patients themselves have to be transported, physicians from other units may need to come for medical consultations, and medical reports have to be written, sent and interpreted. Accordingly, the cooperation between organizational units as well as the medical staff constitutes a crucial task with repetitive, but non-trivial character. In this context, healthcare processes of different complexity and duration can be identified. There are organizational procedures like order entry and result reporting, but also complex and long-running treatment processes like chemotherapy for in- or outpatients.

Physicians have to decide which interventions are necessary or not–under the perspective of costs and invasiveness–or which are even dangerous due to possible side-effects or interactions. Many procedures need preparatory measures of various complexity. Before a surgery may take place, for example, a patient has to undergo numerous preliminary examinations, each of them requiring additional preparations. While some of them are known in advance, others may have to be scheduled dynamically, depending on the individual patient and her state of health, i.e., *looseness* of the overall patient treatment process is a reality.

In general, the tasks of a healthcare process may have to be performed in certain orders, sometimes with complex temporal constraints to be considered [16, 15]. After an injection with contrast medium was given to a patient, for example, some other tests cannot be performed within a certain period of time. In contemporary healthcare environments, physicians still have to coordinate the tasks related to their patients manually, taking into account all the constraints existing in this context. In this context, changing a schedule is not trivial and requires time-consuming communication. For other procedures, medical staff from various departments have to collaborate; i.e., coherent series of appointments have to be arranged and for each activity appropriate information has to be provided. As a drawback, each organizational unit involved in the treatment process of a patient concentrates on the function it has to perform. Thus, the process is subdivided into function- or organization-oriented views, and optimization stops at the border of the department. For all these reasons several problems result. First, patients have to wait, because resources (e.g., physicians, rooms or technical equipment) are not available due to insufficient coordination. Second, medical procedures cannot be performed as planned, if information is missing, preparations are omitted, or a preceding procedure is postponed, canceled or

requires latency time. Depending procedures might then have to be re-scheduled resulting in time-consuming phone calls. Third, if urgently needed results are missing, medical tests or procedures may have to be performed repeatedly causing unnecessary costs and burdening patients.

For all these reasons, from both the patient and the hospital perspective undesired effects occur: Hospital stays can take longer than required and costs or even invasiveness of patient treatment increase. In critical situations, missing information might lead to late or even wrong decisions. Investigations have shown that medical personnel is aware of these problems and that healthcare process support would be highly welcome by medical staff [24]. More and more it is being understood that the correlation between medicine, organization and information is high, and that traditional organizational structures and healthcare information systems only offer sub-optimal support. This even applies more to hospital-wide and cross-hospital processes in health care networks [6].

The roles of physicians and nurses complicate the situation. Both are responsible for many patients and have to provide an optimal treatment process for each of them. Medical tasks are critical to patient care and even minor errors might have disastrous consequences. The working situation is further burdened by frequent context switches. Physicians often work at various sites of a hospital in different roles. In many cases unforeseen events and emergency situations occur, patient status changes, or information necessary to react is missing. Additionally, the physician is confronted with a massive load of data to be structured, intellectually processed, and put into relation to the problems of the individual patient. Typically, physicians tend to make mistakes (e.g., wrong decisions, omission errors) under this data overload.

From the perspective of a patient, a concentration on his treatment process is highly desirable. Similarly, medical staff members wish to treat and help patients and not to spend their time on administrative tasks. From the perspective of healthcare providers, the huge potential of the improvement as well as (semi-)automation of healthcare processes has been identified: length of stay, number of procedures, and number of complications could be reduced. Hence there is a growing interest in process orientation and quality management. Medical and organizational processes are being analyzed, and the role of medical guidelines describing diagnostic and treatment steps for given diagnoses is emphasized [12, 37, 23].

3 Flexibility Needs for Healthcare Processes

Providing appropriate support for the wide range of processes that can be found in healthcare environments (cf. Section 2) poses several challenges. Particularly, flexible process support can be characterized by four major flexibility needs, namely support for variability, looseness, adaptation, and evolution. In the following, a brief summary of each flexibility need is presented and illustrated by a healthcare process scenario.

3.1 Variability

Process variability is characteristic for the healthcare domain and requires healthcare processes to be handled differently—resulting in different *process variants*—depending on the given application context [29, 11]. Typically, process variants share the same core process whereas the concrete course of action fluctuates from variant to variant. Variability in the healthcare services provided, for example, often necessitates support for numerous process variants [11]. Moreover, process variants might exist due to differences in regulations found in different countries or healthcare organizations. Process variability might be further introduced due to different groups of patients, the kind of service provided, peculiarities of the respective service providers, or temporal differences regarding service delivery (e.g., daily changes). In general, the parameters causing process variability are mostly known a priori. Even though the concrete variant can often only be determined during process execution, the course of action for a particular context is well understood.

Example 1. (Process variants for handling medical examinations). Consider the four process variants in Figure 1. The variants have several activities (e.g., `Order Medical Examination`, `Perform Medical Examination`, and `Create Medical Report`) in common. In Figure 1, these common activities are gray-shaded. However, the variants also show differences, e.g., in respect to the kind of examination (i.e., standard vs. emergency medical examination), the way the examination is handled (e.g., scheduling an examination later by making an appointment with the examination unit or registering one for the same day), or the need of specific activities depending on the given application environment (e.g., `Prepare Patient` or `Transport Patient`).

3.2 Adaptation

In general, *process adaptation* represents the ability of a process-aware information system (PAIS) to adapt the process and its structure (i.e., the prespecied process model) to emerging events. Respective events often lead to situations in which the PAIS does not adequately reflect the real-world process anymore. As a consequence, one or several process instances have to be adapted in order to realign the computerized processes with the real-world ones. Note that it is not always possible to predict all exceptional situations and the way they shall be handled during process execution. Even if this had been possible, one would obtain complex and spaghetti-like process models, which are difcult to comprehend and costly to maintain.

Drivers for adaptation. Process adaptations are triggered by different drivers. Adaptations might become necessary to cope with special situations during process execution, which have not been foreseen in the process model, e.g., situations that occur very rarely. Moreover, exceptions occurring in the real-world (e.g., an allergic reaction of a patient) or processing errors (e.g., a failed activity) often require deviations from the standard process.

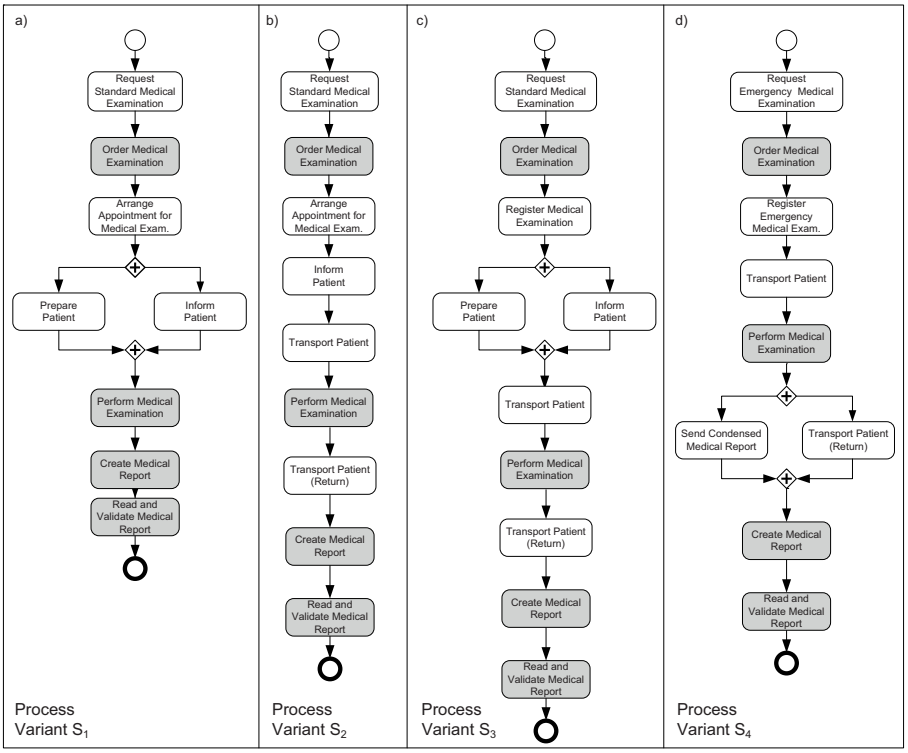

Fig. 1. Examples of healthcare process variants.

Anticipation of adaptation. Many exceptions can be anticipated and, therefore, be planned upfront by capturing them in the process model. Generally, a deviation can only be planned if both the context of its occurrence and measures to handle it are known beforehand. However, it is hardly possible to foresee all exceptions that might occur during the execution of a particular healthcare process. Therefore, support for dealing with *unplanned exceptions* is additionally needed.

Example 2. (Examination procedures in a hospital). A simple examination procedure in a hospital comprises activities like `Enter Order`, `Schedule X-rays`, `Inform Patient`, `Transfer Patient`, `Perform X-rays`, `Create Report`, and `Validate Report`. Even for such a simple process, exceptional situations might occur, which require deviations from the prespecified process. For example, in case of an emergency, there is no time to follow the usual procedure. Instead the patient is immediately examined without making any appointment or preparing the examination facility. To cope with such situation, it should be possible to skip one or more activities. In exceptional situations it can further be required to perform additional (i.e., unplanned) activities for a particular patient (e.g., to carry out an additional preparation activity for the examination). In addition,

changes in appointments, cancelations, and failures in the execution of activities (e.g., omitted preparations, loss of a sample, or incorrect collection of diagnostic material) might lead to deviations from the standard process (e.g., by redoing activities). If an appointment is canceled, for example, the patient treatment process (including the previously made appointment) will have to be aborted.

In summary, in the medical domain, deviations from the standard procedure are rather the norm and have to be flexibly addressed by medical staff.

3.3 Evolution

Evolution represents the ability of the process implemented in a PAIS to change when the corresponding real-world process evolves. As healthcare processes evolve over time, it is not sufficient to implement them once and then to never touch the running PAIS again. In order to ensure that real-world healthcare processes and the PAIS remain aligned, these changes have to be propagated to the PAIS as well. Typically, such evolutionary changes are planned changes at the process type level, which are conducted to accommodate evolving needs.

Drivers for process evolution. In healthcare, process evolution is often driven by changes of medical knowledge, technological changes, and the emergence of new legal constraints. Another driver is organizational learning. All these drivers are external to the PAIS (cf. Fig. 2). In healthcare, the evolution of real-world processes can be triggered by emerging medical knowledge (e.g., new evidence on the effectiveness of a treatment procedure) or changing patient behavior. Changes in the technological context might have far reaching effects on the healthcare processes as well. For example, the increasing popularity of mobile devices is revolutionizing the way how medical staff is interacting with its processes and, hence, the way the process shall be designed [24]. Changes might further be triggered by regulatory adaptations like, for example, the introduction of new laws or clinical practices. Finally, changes of healthcare processes might be a result of organizational learning and be triggered by emerging optimization opportunities or misalignments between real-world healthcare processes and the ones supported by a PAIS.

In addition to external triggers, changes of the processes implemented in a PAIS might become necessary due to developments inside the PAIS, i.e., there exist internal drivers for changes as well [4]. For example, design errors might cause problems during the execution of process instances in the PAIS (e.g., deadlocks or missing data). Moreover, technical problems like performance degradation (e.g., due to an increasing amount of data) may require changes in the PAIS. Finally, poor internal quality of process models (e.g., non-intention revealing naming of activities or redundant process model fragments) might require changes [41].

Extent of evolution. Process evolution may be incremental (i.e., only requiring small changes of the implemented process) as for continuous process improvements, or be revolutionary (i.e., requiring radical changes) as in the context of process innovation or process reengineering.

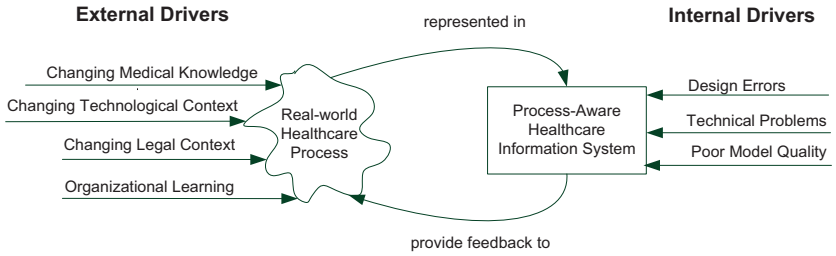

Fig. 2. Drivers for process evolution.

Swiftness of evolution. Depending on the kind of evolutionary change, different requirements regarding the treatment of ongoing process instances exist [27, 32]. In some scenarios, it is sufcient to apply the changes only to those process instances that will be newly created and to complete the ongoing ones according to the old version of the process. This, in turn, would require *deferred evolution* and coexistence of different active versions of a process model within the PAIS. In many practical scenarios, however, *evolutionary changes* have an effect on ongoing process instances as well. For example, regulatory changes often have a retroactive impact and require ongoing process instances, if they have not progressed too far, to be adapted. Such *immediate evolution* is mostly relevant for long-running processes instances, i.e., process instances with a duration up to several weeks or months (e.g., cyclic chemo treatments).

Visibility of evolution. Evolutionary changes may either be changes of the observable process behavior or the internal structure of the PAIS. While changes of the observable behavior are always reflected by the PAIS support of the real-world processes, changes of the internal structure are kept inside the PAIS (e.g., to address poor internal process model quality). Adding activities to a process model (e.g., to add a lab test to a medical procedure for patients being older than 60) constitutes an example of a change concerning the observable behavior. A typical change only affecting the internal structure of the PAIS includes the removal of process model redundancies by extracting common parts to sub-process models [41].

Example 3. (Introduction of new medical devices). The introduction of new medical imaging devices in a hospital might have implications on the corresponding examination process. Assume that due to the high acquisition costs for the new device the hospital decides to use it for examining outpatients as well (in addition to inpatient examinations). This, in turn, implies changes in the registration procedure. These changes not only affect new patients, but ongoing examination processes (i.e., corresponding process instances) as well. In this example, the evolution is triggered through economic concerns. Furthermore, the change is immediate, i.e., it affects ongoing examination processes (i.e., process instances) as well.

3.4 Looseness

Patient treatment processes, which are by nature knowledge-intensive, can be characterized as non-repeatable (i.e., every process instance looks slightly different), unpredictable (i.e., the exact course of action is unknown and is situation-specic), and emergent (i.e., the exact course of action often emerges during process execution when more specific information becomes available). For processes of this category, only their goal is known a priori (e.g., treating the rupture of a patients cruciate ligament). In turn, the parameters determining the exact course of action are typically not known a priori or might change during process execution. As a consequence, such knowledge-intensive processes cannot be fully prespecified. In addition, it is not possible to establish a set of process variants for these processes, since the parameters causing differences between process instances are not known a priori (unlike with variability). Instead, processes of this category require a *loose specication*.

Example 4. (Patient treatment processes). Patient treatment in a hospital usually comprises activities related to patient intake, admission, diagnosis, treatment, and discharge. Typically, treatment processes comprise dozens up to hundreds of activities, and they are long-running (i.e., from a few days to several months). Furthermore, the treatments of two different patients are rarely identical. Instead the course of action often depends on the specic situation like, for example, the health status of the patient, allergies and chemical intolerances, decisions made by the physician, examination results, and clinical indications. This situation may change during the treatment process, i.e., the course of action is unpredictable. Moreover, treatment processes typically unfold during their execution, i.e., examination results yield information determining how to continue with the treatment. The overall treatment process thereby emerges through the arrangement of simple, well-structured processes (e.g., handling medical orders) often resulting in complex process structures.

4 Process Variability Support

As motivated in Section 3.1 and Example 1, respectively, a key flexibility need in healthcare environments is to be able to cope with *process variability*. In general, the reuse of a process model in different application context often results in a large collection of related *process model variants* (*process variants* for short) belonging to the same *process family* [2]. In particular, the process variants pursue the same or similar business objective and have certain activities (and their ordering constraints) in common, while at the same time differences due to their use in different application contexts exist, e.g., certain activities might be only relevant for some of the process variants or different execution paths that need to be taken depending on the application environment.

To properly cope with process variability, a modeling approach for explicitly capturing variability in process models is needed, i.e., a family of related process variants shall be represented in a compact, reusable, and maintainable

manner. Moreover, it should be possible to configure a process family to an individual process variant that fits best to the requirements of the given application context. This way, established practices and process knowledge of a healthcare organization can be reused, while still providing it with the flexibility to individualize its processes to the respective context. Thereby, the selection of the most suitable variant in such an application context is denoted as *process configuration*. For each *conguration option* (e.g., variation point) it must be decided which of the available alternatives shall be chosen. After making these choices, the finally *configured process model* can be transformed into an executable one by dropping those parts that are no longer required. The latter step is called *individualization*. Both the conguration and the individualization of a *configurable process model* constitute design time activities; i.e., they can be accomplished without need for any run-time knowledge.

Existing approaches providing process variability support split the design phase into two sub-phases–one during which the process family is designed, i.e., a *configurable reference process model* and its *configuration options* are specified, and one in which this configurable reference model is configured and individualized for obtaining specific process variants. A more concrete idea of the two phases of a behavior-based approach for capturing the behavior of all process variants in the same artifact (i.e., reference process model) is given in [38]. In this approach, which is denoted as *configurable nodes*, a reference process model merges a multitude of process variants into one congurable model capturing both the commonalities and the differences of the process variants. In respective reference process models, variation points are represented in terms of configurable nodes and execution paths. By conguring these, in turn, the behavior of the reference process model can be customized to the given application context, i.e., a concrete process variant fitting to this context can be derived.

In more detail, in a configurable reference process model, selected activities and control connectors (i.e. gateways) may be flagged as configurable. Such configurable nodes represent variation points of the reference process model and can be associated with a number of configuration alternatives. Furthermore, configuration constraints over the set of configurable nodes may be added to restrict possible combinations of configuration alternatives. By taking a configurable reference process model as input, and setting each of its configurable nodes to exactly one of the allowed alternatives, a particular process variant can be derived.

In principle, any activity or control connector of a reference process model may be flagged as configurable. In the reference process model depicted in Fig. 3, for example, the configurable nodes are highlighted with thicker border. This reference process model describes a family of process variants for managing medical examinations, i.e., for handling medical orders and reporting related results (see Fig. 1 for examples of process variants that may be derived from this configurable model). In detail, the depicted reference process model comprises five configurable activities and eight configurable control connectors. Its non-configurable nodes, in turn, represent the parts common to all process variants. For example,

activity `Perform Medical Examination` denotes such a commonality since it is not configurable. Hence, this activity is contained in all process variants that may be configured out of the reference process model.

In detail, a configurable reference process model may comprise the following configurable elements:

a) **Configurable activities.** There exist three configuration alternatives for a configurable activity: included (ON), excluded (OFF), and conditional (OPT). The first two alternatives allow process engineers to decide at configuration time whether or not to keep an activity in the model of the process variant to be derived. The last alternative allows deferring this decision to the run-time, i.e., the execution of the activity may be dynamically skipped by users depending on the instance-specic context.
b) **Configurable control connectors.** There exist three different kinds of configurable control connectors: Configurable OR, Configurable XOR, and Configurable AND. A configurable control connector may only be configured to a connector being equally or less restrictive, i.e., the derived process model should be able to produce the same or fewer execution traces compared to the original reference process model. To be more precise, a Configurable OR may be configured to a regular OR, or be restricted to an XOR, AND, or just one outgoing/incoming branch. A Configurable XOR, in turn, may be set to a regular XOR or to just one outgoing/incoming branch. Finally, a Configurable AND may only be mapped to a regular AND, i.e., no particular configuration is allowed.
c) **Configuration requirements.** Configuration requirements define constraints over all the configuration alternatives that may be chosen for the configurable nodes of a reference process model. Only if these constraints are met, the resulting process variant is considered as being valid. Configuration guidelines, in turn, do not prescribe mandatory constraints, but only serve as a kind of recommendation guiding users during the configuration. Both configuration requirements and configuration guidelines can be expressed in terms of simple predicates. Graphically, they are depicted as post-it notes attached to one or several configurable nodes.

Example 5. (Configurable reference process model for the handling of medical examinations). Consider the reference process model in Fig. 3. It covers a family of process variants for handling medical examinations, including activities dealing with order handling, scheduling, transportation, and reporting. Examples of process variants that can be configured out of this reference process model are depicted in Fig. 1. The gray-shaded activities in Fig. 3 reflect the common parts of the producible process variants; i.e., these activities are contained in each process variant (see the variant examples in Fig. 1). Process variability, in turn, is caused by varying factors like the kind of examination involved, the way examinations are scheduled, or the decision whether patient transportation is required.

More precisely, emergency and standard medical examinations need to be distinguished from each other (Requirement 1). For standard medical exami-

nations, either an appointment is scheduled or a simple registration is made (Requirement 2). (The latter means, the examination unit is informed about the later arrival of the patient, but does not appoint a date for the examination.) For emergency medical examinations, in turn, a specific registration is needed (Requirement 3). Furthermore, for a standard medical examination, activity `Inform Patient` is always required (Requirement 4). Patient transportation, in turn, is mandatory for emergency medical examinations (Requirement 5), while for standard medical examinations this depends on other domain facts (Guideline 1). A condensed medical report has to be sent in the context of emergency medical examinations to enable quick feedback (Requirement 6). Finally, if the configurable activity `Transport Patient` is switched on, its counterpart (i.e., activity `Transport Patient (Return)`) has to be switched on as well (Requirement 7). Considering all requirements, there exist several activities that may be contained in some process variants, but which are not required for others (e.g., `Prepare Patient` and `Inform Patient`).

Overall, the configurable reference process model from Fig. 3 comprises 5 configurable activities, 8 configurable connectors, 7 configuration requirements, and one configuration guideline. As discussed, configuration requirements constrain the alternatives that may be chosen for the configurable nodes of the reference process model.

Using such a reference process model, the desired process variants can be derived by setting the conguration alternatives of its configurable nodes accordingly (cf. Example 6).

Example 6. (Configuring a reference process model). Consider the four process variants from Fig. 1. The configuration settings needed for deriving the four variants from the given configurable reference process model (cf. Fig. 3) are depicted in Fig. 4. For each process variant, its configuration settings comply with the given configuration requirements, i.e., all four process variants are valid. Note that, in principle, it is not necessary to explicitly specify a configuration alternative for all configurable nodes since these settings can be partially derived from other configuration settings. In Fig. 4, for example, the configuration settings in gray color do not have to be explicitly specified when exploiting the knowledge on the configuration requirements defined in Fig. 3.

As alternative to configurable nodes, the Provop approach [29, 11] provides a structural configuration approach that allows adding, removing or changing process behavior by adjusting the structure of a configurable process model accordingly (e.g., by adding or deleting activities).

Independent of the chosen approach, a particular challenge is to ensure that configured process variants are sound (i.e., correctly executable) and, hence, can be transformed to executable processes (see [39, 10] for corresponding techniques). Not that, when considering the large number of process variants that may be configured out of a reference process model, as well as the many syntactical and semantical constraints these process variants have to obey, this

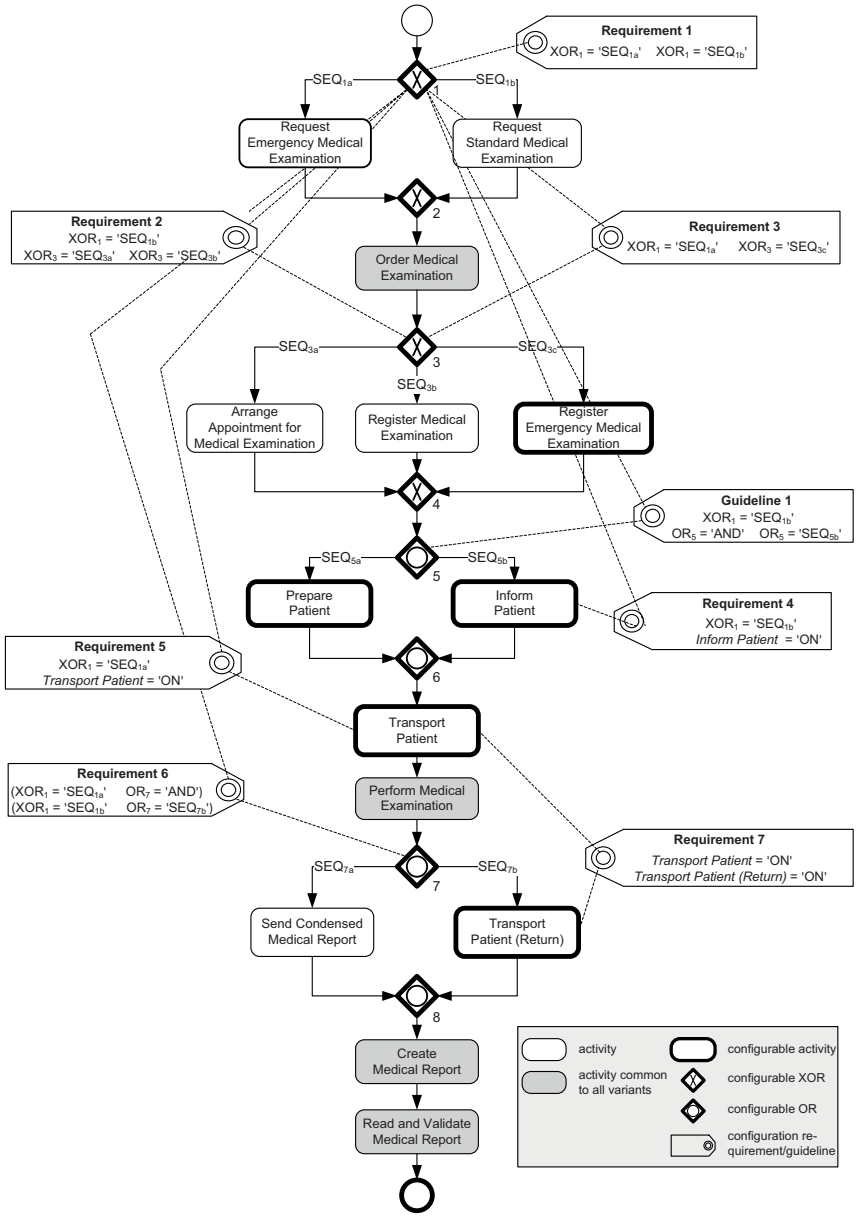

Fig. 3. Example of a configurable reference process model.

constitutes a nontrivial task. Finally, for the above mentioned approaches, high-level configuration user interfaces for domain experts exist, e.g., questionnaire models, feature diagrams, and context-based configurators [36, 32, 9].

	Settings of Configurable Connectors								Settings of Configurable Activities				
	XOR1	XOR2	XOR3	XOR4	OR5	OR6	OR7	OR8	Register Emergency Medical Examination	Prepare Patient	Inform Patient	Transport Patient	Transport Patient (Return)
Process variant S1	**SEQ1b**	SEQ1b	**SEQ3a**	SEQ3a	**AND**	AND	**SEQ7b**	SEQ7b	OFF	**ON**	ON	**OFF**	**OFF**
Process variant S2	**SEQ1b**	SEQ1b	**SEQ3a**	SEQ3a	**SEQ5b**	SEQ5b	**SEQ7b**	SEQ7b	OFF	**OFF**	ON	**ON**	ON
Process variant S3	**SEQ1b**	SEQ1b	**SEQ3b**	SEQ3b	**AND**	AND	**SEQ7b**	SEQ7b	OFF	**ON**	ON	**ON**	ON
Process variant S4	**SEQ1a**	SEQ1a	**SEQ3c**	SEQ3c	**SEQ5b**	SEQ5b	**AND**	AND	**ON**	**OFF**	**OFF**	**ON**	ON

Fig. 4. Examples of configuration settings.

Altogether, enhancing process-aware healthcare information systems with configurable reference process models as well as the capability to derive sound process variants from them, will foster the reuse of process knowledge and increase process model quality in large process repositories

5 Process Adaptation Support

As discussed in Section 3.2, in general, it is not possible to anticipate all exceptions in a healthcare environment and to capture their handling in a prespecified process model at design time. Hence, authorized process participants [44] should be allowed to situationally adapt single process instances running in the PAIS to cope with the non-anticipated exceptions and to realign the digital process running in the PAIS with the real-world case; e.g., by inserting, deleting, or moving activities for one specific process instance. Providing PAIS support for such instance-specific deviations from a prespecified process model, however, must not shift the responsibility for ensuring PAIS robustness to end-users. Instead, the PAIS must provide comprehensive support for the correct, secure, and robust handling of run-time exceptions through ad-hoc process instance changes.

To cope with unanticipated exceptions, authorized users shall be allowed to delete activities, to postpone their execution, to bring the execution of activities forward even though their preconditions have not yet been met, or to add activities not considered in the process model so far [28]. Generally, such behavioral changes of a process instance require structural adaptations of the corresponding process model, which shall solely be applied to that particular process instance. Examples of structural adaptations include the insertion, deletion, or movement of activities and process fragments respectively. While movements change activity positions, and thus the structure of a process model, insertions and deletions additionally modify the set of activities contained in a process model. In this

context, adaptive process management technologies like ADEPT [4, 27, 31] provide high-level change operations, e.g., to move an activity or an entire process fragment within a process model. Usually, the change operations abstract from the concrete process model transformations to be conducted, i.e., instead of specifying a set of change primitives, the user applies one or more high-level change operations to realize the desired process model adaptation. ADEPT associates pre- and post-conditions with the high-level change operations in order to guarantee model correctness after each adaptation, i.e., to ensure correctness by construction [4]. A comprehensive set of change patterns, which are useful for structurally adapting processes models and, hence, process model behavior can be found in [42].

Example 7. (Structural adaptations of a process model). Figure 5 depicts a simple example of a structural process model adaptation referring to a very simplied patient treatment process. As illustrated in Fig. 5a, usually, the treatment process starts with the admission of the patient to the hospital. After having registered the patient, he is treated by a physician. Finally, an invoice for the treatment provided is created. Assuming that a particular patient is in a critical condition, it might become necessary to deviate from the prespecified process model to handle this exception; the treatment of the patient might have to start right away, performing the necessary steps for his registration at a later stage. To capture this behavior in the model of the respective process instance, activity `Treat Patient` has to be arranged in parallel with activity `Register Patient` (cf. Figure 5b), i.e., the unanticipated exception is handled by restructuring the model driving the execution of the respective process instance.

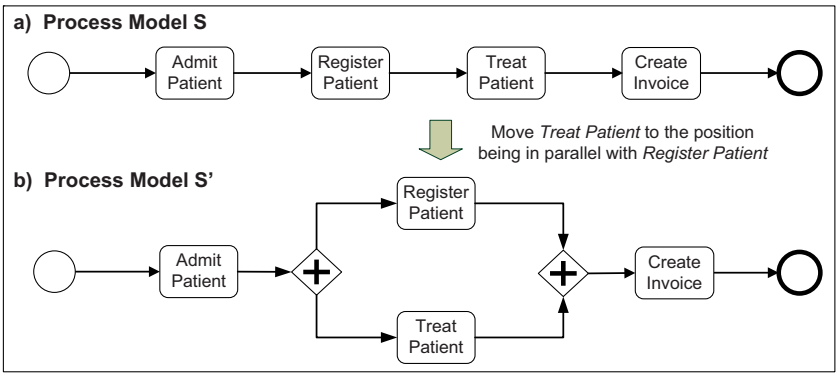

Fig. 5. Example of a structural process adaptation

To correctly deal with ad-hoc changes, process instance states need to be taken into account as well. Generally, the applicability of a particular ad-hoc change depends on the state of the respective process instance. Example 8 illustrates this.

Example 8. (Ad-hoc changes of healthcare process instances). Consider process model S on the left hand side of Fig. 6a. Assume that S is transformed into a correct process model S' by adding two activities (i.e., `Test for Allergies` and `Deliver Drug`) as well as a data dependency between them; i.e., `Test for Allergies` writes data object `Allergy Record`, which is then read by `Deliver Drug`. Assume further that this structural model change shall be applied to the process instances depicted in Fig. 6b and currently being executed according to process model S. Regarding instance I_1 the described change can be applied without any problem as its execution has not yet entered the change region (cf. Fig. 6c). Changing instance I_2 in an uncontrolled manner, however, would result in an inconsistent process instance state; i.e., activity `Prepare Patient` would be running even though its predecessor, activity `Test for Allergies`, would not have been completed. As a consequence, `Deliver Drug` might be invoked accessing data element `Allergy Record` even though this data element might not have been previously written. Regarding instance I_3, the described change may be applied. However, when relinking the execution of I_3 to S', activity `Prepare Patient` needs to be disabled and corresponding work items be withdrawn from user worklists. Additionally, the newly inserted activity `Test for Allergies` has to be enabled.

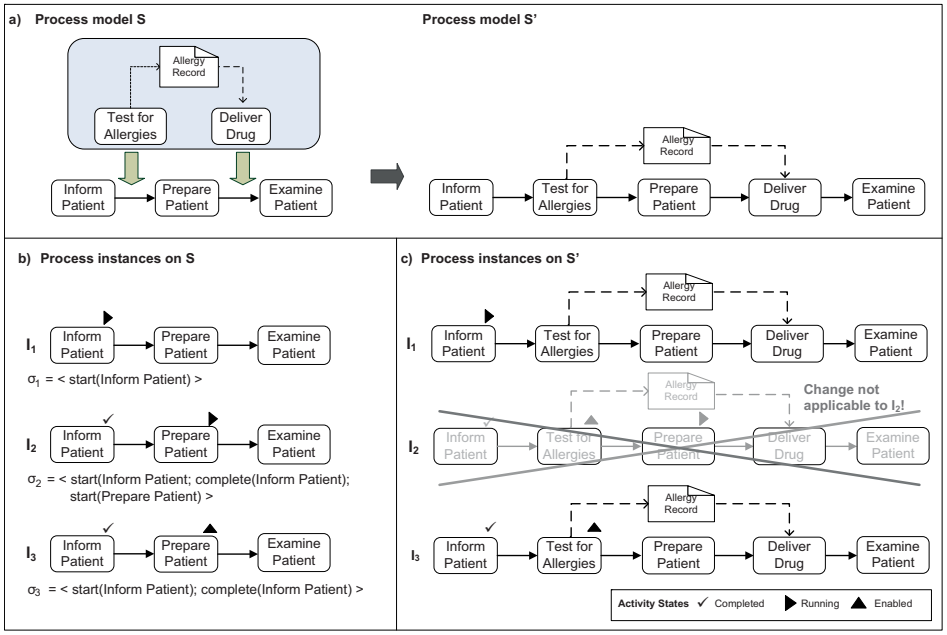

Fig. 6. State-compliant adaptation of process instances.

As illustrated by Example 8, structural changes of a process instance require adaptations of the process instance state (i.e., the states of the corresponding activities) as well. Generally, the respective state adaptations depend on the applied process model change (e.g., deleting a process fragment vs. adding one) as well as on the current state of the process instance. Depending on the position where an activity is inserted, for example, it might become necessary to immediately enable the inserted activity or to disable other ones before continuing with the execution of the process instance. By contrast, when changing a not yet entered region of a process instance, no state adaptations become necessary.

In order to provide advanced user support, end-users should be supported in reusing knowledge about ad-hoc changes, which were previously applied to other process instances in a similar problem context. Accordingly, the changes must be recorded by the PAIS and be annotated with contextual information (e.g., reasons of the ad-hoc change). The latter, in turn, is needed to be able to present knowledge about those previous ad-hoc changes to the user being relevant in the current exceptional situation. For example, an MRT must not be skipped for patients in general, but for those having a cardiac pacemaker.

An approach that facilitates ad-hoc changes of process instances during runtime by supporting the retention and reuse of previously applied instance changes is presented in [45, 43]. In particular, this approach automates change retrieval by considering structured information about the current application context; e.g., the occurred exception and the current state of the process instance to be adapted. Further, if ad-hoc changes applied in a similar context can be retrieved in the given exceptional situations, but cannot be reused directly (e.g., in case the process instance has progressed beyond the point that the ad-hoc change can be directly applied), user support for adapting the respective change definition to the situation at hand is provided.

In summary, this section emphasized the need for structurally adapting the process model of single process instances during run-time in order to cope with unanticipated exceptions. We discussed fundamental issues that emerge due to ad hoc changes and showed how they can be addressed by adaptive PAISs. The section referred to high-level process adaptation patterns for defining ad-hoc changes at an abstract level (e.g., to move an activity). Additionally, it discussed the importance of considering the state of process instances as well as to adapt it when applying ad-hoc changes. In this context, we emphasized that a particular process instance only then might be dynamically changed, if the current instance state complies with the resulting process model (i.e. *state compliance*). We further discussed how users may be supported in reusing knowledge about previous ad-hoc changes applied in similar exceptional situations.

6 Process Evolution Support

As discussed in Section 3.3, any process-aware information system run in a healthcare environment should be able to cope with evolutionary process changes. This section presents fundamental techniques to cope with the evolution of

healthcare processes as implemented in a PAIS at a technical level, i.e., to realize respective process changes within the PAIS. The basic assumption is that the healthcare processes are represented by prespecified process models in the PAIS, and changes of the real-world healthcare process require the corresponding process models to evolve accordingly at the implementation level. A major challenge in this context concerns the handling of long-running process instances that were created based on the old process model, but are now required to comply with a new specication (i.e., a new model version) and, therefore, shall be migrated to it [30, 35]. As thousands of active process instances might be affected, accomplishing such a migration correctly and efficiently becomes crucial [34].

6.1 Deferred Process Evolution

When evolving a process model S to a new process model version S' at the process type level, the PAIS must properly deal with corresponding process instances, i.e., process instances that were started and partially executed on S, but have not been completed yet. The easiest way to properly complete these running process instances is to continue their execution based on the original process model S, whereas new process instances may be created and executed based on the new model version S'—this approach is denoted as *deferred process model evolution* in [32]. In particular, it requires support for version control as well as for the coexistence of process instances belonging to different process model versions of a particular process type.

6.2 Immediate Process Evolution and Instance Migration

While the coexistence of process instances running on different process model versions is sufficient to support deferred evolution, long-running process instances often require immediate evolution, i.e., these process instances shall be migrated on-the-fly to the new process model version if possible. Example 9 illustrates this need.

Example 9. (Need for immediate process model evolution and process instance migration). Consider a patient treatment process and assume that due to newly emerging legal requirements patients have to be informed about certain risks before a specific surgery may take place. Assume further that this change is also relevant for patients whose treatment process was already started. If the respective treatment process is supported by a PAIS, stopping all ongoing process instances (i.e., treatments), aborting them, and restarting them does not constitute a viable option. As a large number of treatment processes might be concurrently running, applying this change manually to the instances of ongoing treatment processes in the PAIS is hardly a realistic option. Instead, PAIS support is needed to add this new activity to all patient treatment processes for which this is still feasible, e.g., for which the surgery has not been started or completed yet.

As a particular challenge, *immediate process instance migrations* have to be accomplished in a controlled manner, i.e., none of the correctness properties (e.g., soundness) guaranteed through the verification of a process model at design time must be violated for any of the migrated process instances. If this cannot be guaranteed for a particular process instance, it must not be migrated, but remain running on the old process model version. To meet this goal, it first has to be ensured that the new process model version S' is correct; i.e., S' has to satisfy the syntactical and structural properties of the process modeling language (e.g., BPMN 2.0) used, and it further must constitute a sound (i.e., correctly executable) process model.

The problem here is the same as when applying an ad-hoc change to a single process instance at run-time (cf. Section 5); i.e., similar challenges exist as for ad-hoc changes. In particular, the state of the process instances to be migrated (i.e., their execution traces) must be taken into account when deciding on whether their execution may be relinked from a process model S to a new model version S' (i.e., whether the instances may migrate to S'). A widespread correctness notion used for deciding about whether or not a particular process instance may be dynamically migrated to a new process model version S' is *state compliance*—a process instance I is denoted as being *state compliant* with an updated process model S' and can therefore be migrated to it, if the execution trace of I, which records all execution events related to I, is producible on S' as well. Using this correctness notion in the context of process model evolution, it can be ensured that process instances whose state has progressed too far will not be migrated to the new process model version S', i.e., they will remain running on the original process model version. Furthermore, when migrating a running process instance to a new process model version its state has to be automatically adapted. For example, an already enabled activity may have to be disabled when inserting an activity directly preceding it or a newly added activity may have to be immediately enabled if the preconditions for its execution are met.

Example 10 illustrates a process model evolution together with the controlled migration of related process instances. Note that this example is similar to the healthcare scenario discussed in the context of Example 8.

Example 10. (Controlled process instance migration). Consider the evolution of process model S to S' as depicted at the top of Figure 7. Furthermore, consider the three process instances I_1, I_2, and I_3 now running on S. Only those process instances (i.e., I_1 and I_2) are migrated to the new process model S', which are state compliant with it: I_1 can be migrated to S' without need for any instance state adaptation. Furthermore, I_2 can be migrated to S' as well. However, in this case the newly inserted activity `X` becomes immediately enabled, whereas the already enabled activity `B` becomes disabled. Finally, process instance I_3 cannot be migrated to S', as it is not state compliant with this model. Hence, I_3 remains running on the original process model S.

Note that the controlled evolution of process instances as illustrated in Example 10 requires support for the coexistence of process instances running on

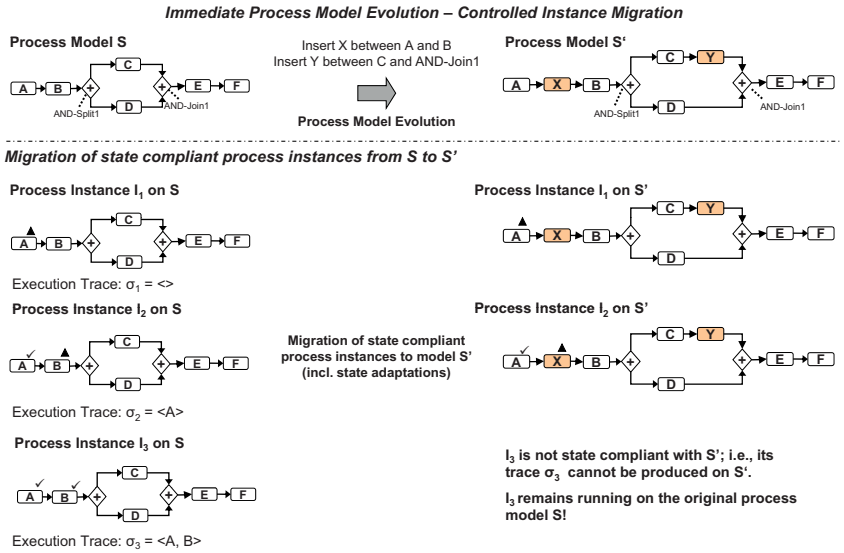

Fig. 7. Process model evolution and process instance migration

different versions of a particular process model, as well as the use of appropriate correctness notions for deciding whether or not process instances can be correctly executed on the new model version.

7 Process Looseness Support

As motivated in Section 3.4, in the healthcare domain, it is not always possible to fully prespecify the model of a healthcare process in advance, i.e., while parts of the respective process model are known at design time, others might be uncertain and can solely be specified during process execution. For example, the treatment of a particular patient depends on his actual physical data and the list of symptoms and medical problems reported during process execution. To cope with this uncertainty, decisions regarding the exact specification of selected parts of the process model may be deferred to the run-time, i.e., instead of requiring the process model to be fully specified prior to the creation and execution of corresponding process instances, parts of the model can remain unspecified. Process participants then may add information regarding the unspecified parts of the process model during process execution.

This section presents two *decision deferral patterns*, which can be also applied to healthcare processes, i.e., *Late Selection* and *Late Modeling & Composition*. As opposed to structural process adaptations (cf. Section 5), whose application is not restricted a priori to a particular process model part, the decision deferral patterns define constraints concerning the parts of a process model that may be changed or expanded. In particular, the application of the patterns has to be

anticipated at design time, which is accomplished by defining regions in the process model where potential changes may be performed during run-time (decision deferral patterns are therefore also denoted as *patterns for changes in predefined regions* in [42]).

A *loosely specified process* is therefore defined by a process model, which is not fully prespecified, but keeps some parts unspecified at design time by deferring decisions to the run-time. The aforementioned patterns differ in the degree of freedom provided to the user and the planning approach employed when concretizing the loosely specified parts of the process model during run-time. Moreover, the scope of decision deferral (i.e., prespecified parts of the process model or entire process) has to be considered. Taken together, these dimensions determine the provided degree of looseness. The considered patterns are as follows:

Late selection of process fragments. This pattern allows deferring the selection of the implementation of a particular process activity to the run-time. At design time, solely a *placeholder activity* has to be provided. Its concrete implementation is then selected during run-time among a predefined set of alternative process fragments either based on defined rules or on user decisions (cf. Fig. 8). However, the selection must be accomplished before the placeholder activity is enabled or when it becomes enabled. Finally, the fragment substituting the placeholder activity may either be an atomic activity or a sub-process.

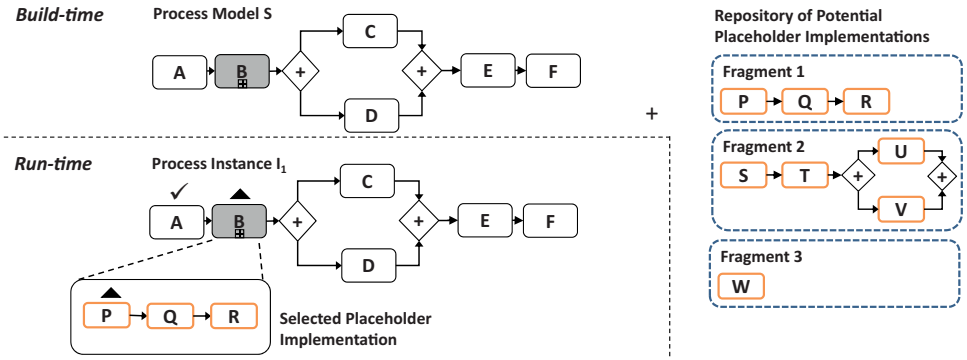

Fig. 8. Late selection of process fragments

Late modeling & composition of process fragments. This pattern offers more freedom compared to Late Selection. It allows for the on-the-fly modeling of selected parts of the process model at run-time, i.e., at design time, only a placeholder activity is provided, whose implementation is then provided during run-time (cf. Fig. 9). Building blocks that may be used for late modeling & composition can either be all process fragments from a repository, a constraint-based subset of the fragments from the repository, or newly defined activities or process fragments. In this context, constraints may be defined, which have

to be considered when modeling or composing an unspecified process part. Furthermore, late modeling can take place upon creation of the process instance, or when the placeholder activity becomes enabled or a particular state in the process is reached. Depending on the pattern variant users start late modeling with an empty template or take a predefined template as a starting point and adapt it as required.

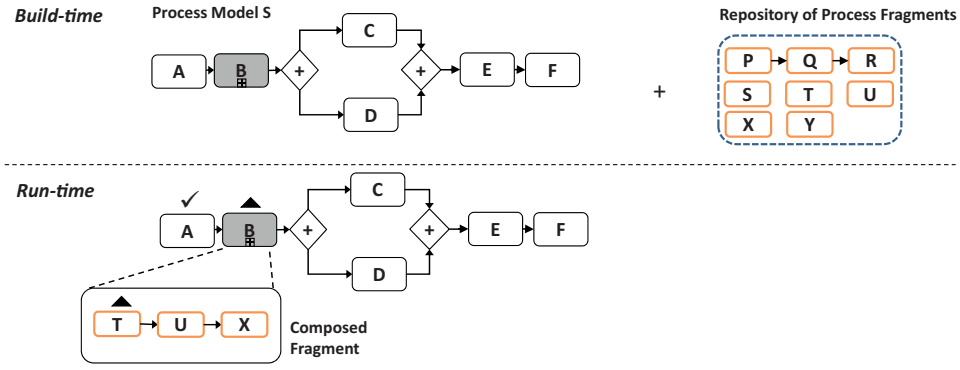

Fig. 9. Late modeling of process fragments

To give an idea of how decision deferral patterns can be implemented and applied in a healthcare context, with Worklets [1] we present a concrete approach realizing the Late Selection pattern. For this, each activity is associated with a set of sub-process fragments, which may be dynamically extended (i.e., additional fragments can be added on the fly)(cf. Figure 10). Again, the activities of a sub-process may be linked with a set of fragments. During run-time choices are made dynamically out of the set of subprocess fragments when activities become enabled. The selection of a suitable fragment is made using hierarchically organized selection rules–called ripple down rules. Users may adjust the automatic choice by adding selection rules. Once a fragment has been chosen, the placeholder activity is replaced by it.

Example 11. (Late selection with Worklets). Fig. 10 illustrates the Worklet approach using a simplified example from the healthcare domain. The prespecified process model consists of the four activities `Admit Patient`, `Perform Triage`, `Treat Patient`, and `Discharge Patient`. Activity `Treat Patient` is linked with a set of 7 subprocesses. Depending on the actual physical condition of the patient and his list of symptoms, a suitable treatment is chosen during runtime. For this, the ripple down rules are evaluated once activity `Treat Patient` becomes enabled. The evaluation of the rules starts with the root node which always evaluates to true. As the next step, condition `Fever = True` is evaluated. If this condition holds subprocess `Treat Fever` is selected and activity `Treat Patient` is replaced by it. Otherwise, the evaluation continues with the next rule (i.e., condition `Wound = True`).

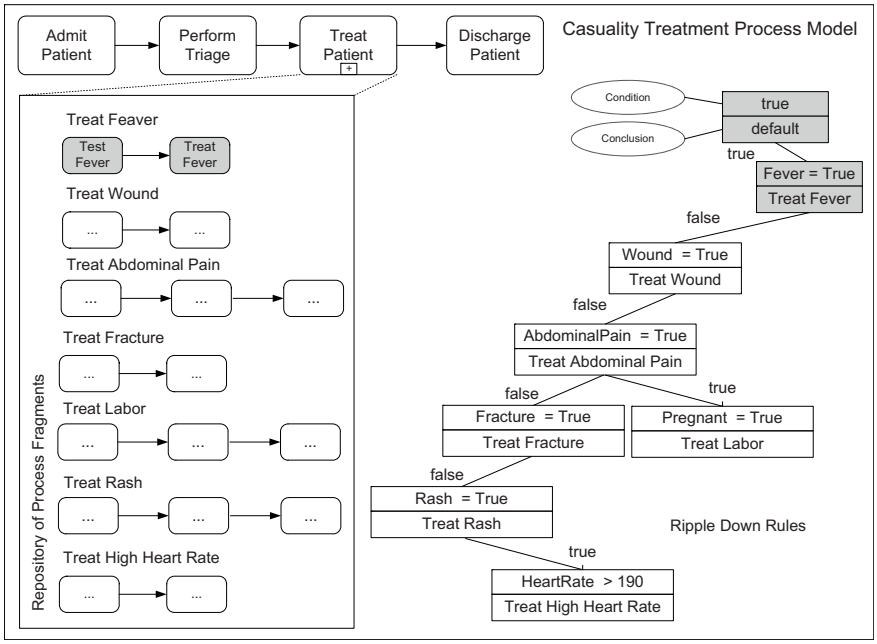

Fig. 10. Late selection with Worklets (adopted from [1]).

A similar approach like Worklets, which is called Context-aware Process Injection (CaPI), is described in [20].

8 Other Process Flexibility Approaches

For many years, the BPM community has recognized that a PAIS needs to be able to cope with real-world exceptions, uncertainty, and evolving processes [32]. To address the flexibility needs discussed in Section 3, besides the concepts and techniques presented in the previous sections, a variety of other process support paradigms, including *case handling*, *constraint-based processes*, and *data- and object-centric processes*, have been suggested and applied to healthcare scenarios.

8.1 Constraint-based Processes

This sub-section introduces constraint-based approaches to process modeling and execution, which enable loosely specified processes as well [40, 8]. While prespecified process models define *how* things have to be done (i.e., in what order and under what conditions activities shall be executed), *constraint-based process models* focus on *what* should be done by describing the activities that may be performed and the constraints prohibiting undesired execution behavior.

Example 12 deals with a simplied medical guideline we adopted from [40]. It describes a constraint-based process of treating a patient admitted to the emergency room of a hospital suspected of having a fracture (cf. Figure 11).

Example 12. (Fracture treatment process). Consider Figure 11. Before any treatment may be chosen, activity `Examine Patient` has to be performed by a physician (constraint `init`). If required, additional medical diagnosis is done by executing activity `Perform X-rays`. Depending on the presence and type of fracture, four different treatments exist: `Prescribe Sling`, `Prescribe Fixation`, `Perform Surgery`, and `Apply Cast`. Except for `Apply Cast` and `Prescribe Fixation`, which are mutually exclusive (constraint `not co-existent`), the treatments can be applied in any combination and each patient receives at least one of them (`1-of-4 constraint`). Activity `Perform X-rays` is not required if the specialist diagnoses the absence of a fracture when performing activity `Examine Patient`. If activity `Perform X-rays` is omitted, only the treatment `Prescribe Sling` may be applied. All other treatments require `Perform X-rays` as preceding activity in order to rule out the presence of a fracture, or to decide how to treat it (constraint `precedence`). Simple fractures can be treated just by performing activity `Apply Cast`. For unstable fractures, in turn, activity `Prescribe Fixation` may be preferred over activity `Apply Cast`. When performing activity `Perform Surgery`, the physician is further advised to (optionally) execute activity `Prescribe Rehabilitation` afterwards (optional constraint `response`). Moreover, the physician may execute activity `Prescribe Medication` (e.g., pain killers or anticoagulants) at any stage of the treatment. Note that activities `Examine Patient` and `Perform X-rays` may be also performed during treatment.

Altogether, the process of treating a fracture comprises the activities `Examine Patient`, `Perform X-rays`, `Prescribe Sling`, `Prescribe Fixation`, `Perform Surgery`, `Apply Cast`, `Prescribe Rehabilitation`, and `Prescribe Medication`. Moreover, constraints prohibit undesired execution behavior, e.g.:

1) Activity Examine Patient has to be executed first.
2) Each patient gets at least one out of four treatments (i.e., `Prescribe Sling`, `Prescribe Fixation`, `Perform Surgery`, or `Apply Cast`).
3) Activities `Apply Cast` and `Prescribe Fixation` are mutually exclusive.
   – `Perform X-rays` is a prerequisite for all treatments except `Prescribe Sling`.
4) If activity `Perform Surgery` is performed for a certain patient, the physician will be advised to execute activity `Prescribe Rehabilitation` afterwards.

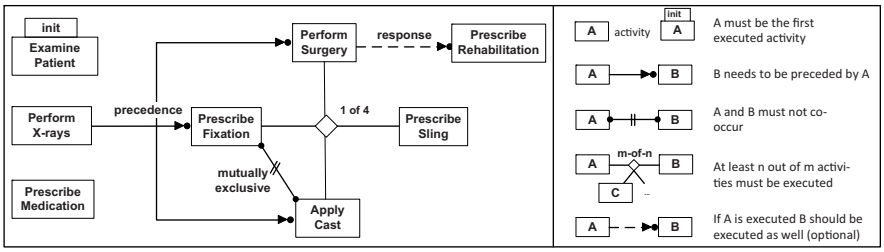

Fig. 11. Example of a constraint-based process model.

Figure 11 depicts the loosely specified process model corresponding to Example 12 when using a constraint-based process modeling approach. The boxes represent activities and the relations between them are different kinds of constraints for executing these activities. The depicted model contains mandatory constraints (solid lines) as well as one optional constraint (dashed line). As opposed to fully prespecified process models that describe how things have to be done, constraint-based process models focus on the logic that governs the interplay of actions in the process by describing the activities that can be performed and those constraints prohibiting undesired behavior.

Note that in more complex cases, the physician in charge may have to choose from dozens or even hundreds of activities. While some of them may be executed any number of times and at any point in time during the treatment process, for others a number of constraints have to be obeyed; e.g., certain activities may have to be preceded or succeeded by other activities or may even exclude certain activities. Moreover, depending on the particular patient and his medical problems, certain activities might be contraindicated and should therefore not be chosen. The challenge is to provide PAIS support for such knowledge-intensive processes and to seamlessly integrate the described constraints within the physicians work practice. Generally, the structure of *knowledge-intensive processes* strongly depends on user decisions made during process execution; i.e., it dynamically evolves.

8.2 Object-centric Processes

The process flexibility approaches presented in the previous sections are *activity-centric*, i.e., they focus on the coordinated execution of a set of business functions, represented by atomic process steps (i.e., activities), as well as the control and data flow between them. Typically, the primary drivers of *activity-centric processes* are the *events* related to activity completions. In turn, business data is rather "unknown" to the process engine of an activity-centric PAIS. The latter only maintains simple data elements needed for control flow routing and for assigning values to activity input parameters. In particular, business objects and their attributes are usually outside the control of an activity-centric PAIS [14].

In healthcare, however, one can also find processes not being activity-centric, but whose execution is driven by user decisions and patient data [3]. These processes are usually unstructured or semi-structured, and tend to be knowledge-intensive. In particular, they can not be straight-jacketed into a set of activities with prespecified precedence relations (as in the examples presented above). As a consequence, the activity-centric approaches presented so far do not adequately support these processes [14]. Moreover, the primary driver for the progress of a process is not the event related to activity completion, but the availability of certain values for data objects. When implementing such *user- and data-driven processes* in a PAIS, a tight integration of processes, data, and users becomes necessary [26].

There exists pioneering work targeting at user- and data-driven process management and enabling such a tight integration. As a first example, the case han-

dling paradigm [7] needs to be mentioned. It focuses on the case (e.g., a patient treatment) and its flexible handling, whereby the progress of a case is determined by the values of its data objects, i.e., case execution is *data-driven*.

While case handling is appropriate for supporting simple process scenarios, it does not provide sufficient abstractions to deal with more complex and inter-dependent cases. Approaches focusing on *object-aware processes* offer more promising perspectives in this context; i.e., here the PAIS manages data by means of object types (e.g., medical report, medical examination) that comprise object attributes and relations to other object types. Accordingly, a business process coordinates the processing of several business objects of the same or different type among end-users enabling them to cooperate and communicate with each other. As shown in [13], object-aware processes provide a high degree of abstraction by enabling two levels of process granularity: *object behavior* and *object interactions*. Furthermore, object-aware process management supports *data-driven process execution*, *exible choice of activity granularities*, and *integrated access to business processes and business data*. In [13] a framework realizing flexible object-aware process support based on a tight integration of processes, functions, data, and users is presented. In particular, the framework provides support for coordinating the execution of related processes and the interactions of their corresponding objects. In turn, the application of this framework to sophisticated healthcare scenarios is presented in [3].

9 Summary

When efforts are taken to improve and automate healthcare processes through the introduction of a PAIS, it is of utterly importance that this does not lead to rigidity. Otherwise, the PAIS will not be accepted by clinical staff. Furthermore, variability in healthcare processes is deeply inherent to the medical domain, and unforeseen events are to some degree a normal phenomenon in current practice. PAISs should therefore enable a high degree of flexibility throughout the entire process life cycle.

To enable the required process flexibility in healthcare environments, several challenges need to be tackled: First, variability in healthcare processes, which is known prior to their implementation, should be captured and made known to the PAIS. Second, authorized process participants should be free to react in unplanned or exceptional situations by gaining complete initiative and by deviating from the prespecified process whenever required. Note that in the healthcare domain the process participants are usually trained to do so and, hence, enabling ad-hoc deviations from the prespecified process model forms a key part of process flexibility. In all these scenarios, the PAIS should be easy to handle, self-explaining, and—most important—its use should be not more cumbersome and time-consuming than simply handling the unplanned situation or exception by a telephone call to the right person. Third, process models may evolve over time due to environmental changes (e.g., process redesign or new laws). Consequently, a PAIS should support process model evolution and provide

appropriate techniques for dealing with already running process instances in this context. Flexibility features of a PAIS must neither affect its robustness nor the correct execution of the healthcare processes it implements. Fourth, to support knowledge-intensive processes, PAISs should enable the loose specification of process models at design time and their renement during run-time, as well as data- and user-driven processes in cases where activity-centric approaches do not fit at all.

Existing approaches for the flexible support of prespecified or loosely specified processes have been already established in industrial practice for several years. Hence, they provide a rather high degree of maturity. By contrast, approaches enabling knowledge-intensive processes constitute cutting-edge research, but will become more mature and emerge in practical settings in a few years. While the conceptual and theoretical foundations of the different paradigms are well understood, there still exist numerous challenges regarding their practical use in healthcare environments. Amongst others, these challenges include proper end-user assistance, flexible support of mobile healthcare processes, and flexibility in cross-organizational processes (e.g., in the context of healthcare networks).

References

1. M. Adams, A. H. M. ter Hofstede, D. Edmond, and W. M. P. van der Aalst. Worklets: A service-oriented implementation of dynamic flexibility in workflows. In *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE, OTM Confederated International Conferences, CoopIS, DOA, GADA, and ODBASE 2006, Montpellier, France, October 29 - November 3, 2006. Proceedings, Part I*, pages 291–308, 2006.
2. C. Ayora, V. Torres, B. Weber, M. Reichert, and V. Pelechano. VIVACE: A framework for the systematic evaluation of variability support in process-aware information systems. *Information & Software Technology*, 57:248–276, 2015.
3. C. M. Chiao, V. Künzle, and M. Reichert. Object-aware process support in healthcare information systems: requirements, conceptual framework and examples. *Int'l Journal on Advances in Life Sciences*, 5(1 & 2):11–26, July 2013.
4. P. Dadam and M. Reichert. The ADEPT project: a decade of research and development for robust and flexible process support. *Computer Science - R&D*, 23(2):81–97, 2009.
5. P. Dadam, M. Reichert, and K. Kuhn. Clinical workflows - the killer application for process-oriented information systems? In *Proc. 4th Int'l Conf. on Business Information Systems (BIS'00)*, pages 36–59. Springer, April 2000.
6. W. Fdhila, C. Indiono, S. Rinderle-Ma, and M. Reichert. Dealing with change in process choreographies: Design and implementation of propagation algorithms. *Information Systems*, 49:1–24, 2015.
7. C. W. Günther, M. Reichert, and W. M. P. van der Aalst. Supporting flexible processes with adaptive workflow and case handling. In *17th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, WETICE 2008, Rome, Italy, June 23-25, 2008, Proceedings*, pages 229–234, 2008.
8. C. Haisjackl, I. Barba, S. Zugal, P. Soffer, I. Hadar, M. Reichert, J. Pinggera, and B. Weber. Understanding declare models: strategies, pitfalls, empirical results. *Software and System Modeling*, 15(2):325–352, 2016.

9. A. Hallerbach, T. Bauer, and M. Reichert. Context-based configuration of process variants. In *3rd International Workshop on Technologies for Context-Aware Business Process Management (TCoB 2008)*, pages 31–40, June 2008.
10. A. Hallerbach, T. Bauer, and M. Reichert. Guaranteeing soundness of configurable process variants in Provop. In *2009 IEEE Conference on Commerce and Enterprise Computing, CEC 2009, Vienna, Austria, July 20-23, 2009*, pages 98–105, 2009.
11. A. Hallerbach, T. Bauer, and M. Reichert. Capturing variability in business process models: the Provop approach. *Journal of Software Maintenance*, 22(6-7):519–546, 2010.
12. E. Kilsdonk, L. W. P. Peute, and M. W. M. Jaspers. Factors influencing implementation success of guideline-based clinical decision support systems: A systematic review and gaps analysis. *Int J Medical Informatics*, 98:56–64, 2017.
13. V. Künzle and M. Reichert. Philharmonicflows: towards a framework for object-aware process management. *Journal of Software Maintenance*, 23(4):205–244, 2011.
14. V. Künzle, B. Weber, and M. Reichert. Object-aware business processes: fundamental requirements and their support in existing approaches. *International Journal of Information System Modeling and Design*, 2(2):19–46, 2011.
15. A. Lanz, M. Reichert, and B. Weber. Process time patterns: A formal foundation. *Information Systems*, 57:38–68, 2016.
16. A. Lanz, B. Weber, and M. Reichert. Time patterns for process-aware information systems. *Requirements Engineering*, 19(2):113–141, 2014.
17. R. Lenz and M. Reichert. IT support for healthcare processes - premises, challenges, perspectives. *Data Knowledge Engineering*, 61(1):39–58, 2007.
18. C. Li, M. Reichert, and A. Wombacher. Mining business process variants: challenges, scenarios, algorithms. *Data Knowledge Engineering*, 70(5):409–434, 2011.
19. N. Mundbrod, F. Beuter, and M. Reichert. Supporting knowledge-intensive processes through integrated task lifecycle support. In *19th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2015, Adelaide, Australia, September 21-25, 2015*, pages 19–28, 2015.
20. N. Mundbrod, G. Grambow, J. Kolb, and M. Reichert. Context-aware process injection - enhancing process flexibility by late extension of process instances. In *On the Move to Meaningful Internet Systems: OTM 2015 Conferences - Confederated International Conferences: CoopIS, ODBASE, and C&TC 2015, Rhodes, Greece, October 26-30, 2015, Proceedings*, pages 127–145, 2015.
21. N. Mundbrod, J. Kolb, and M. Reichert. Towards a system support of collaborative knowledge work. In *Business Process Management Workshops - BPM 2012 International Workshops, Tallinn, Estonia, September 3, 2012. Revised Papers*, pages 31–42, 2012.
22. M. Peleg, J. Somekh, and D. Dori. A methodology for eliciting and modeling exceptions. *Journal of Biomedical Informatics*, 42(4):736–747, 2009.
23. M. Peleg and S. W. Tu. Design patterns for clinical guidelines. *Artificial Intelligence in Medicine*, 47(1):1–24, 2009.
24. R. Pryss, N. Mundbrod, D. Langer, and M. Reichert. Supporting medical ward rounds through mobile task and process management. *Information Systems E-Business Management*, 13(1):107–146, 2015.
25. M. Reichert. What BPM technology can do for healthcare process support. In *Artificial Intelligence in Medicine - 13th Conference on Artificial Intelligence in Medicine, AIME 2011, Bled, Slovenia, July 2-6, 2011. Proceedings*, pages 2–13, 2011.

26. M. Reichert. Process and data: Two sides of the same coin? In *On the Move to Meaningful Internet Systems: OTM 2012, Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2012, Rome, Italy, September 10-14, 2012. Proceedings, Part I*, pages 2–19, 2012.
27. M. Reichert and P. Dadam. Enabling adaptive process-aware information systems with ADEPT2. In J. Cardoso and W. van der Aalst, editors, *Handbook of Research on Business Process Modeling*, pages 173–203. Information Science Reference, Hershey, New York, March 2009.
28. M. Reichert, P. Dadam, and T. Bauer. Dealing with forward and backward jumps in workflow management systems. *Software and System Modeling*, 2(1):37–58, 2003.
29. M. Reichert, A. Hallerbach, and T. Bauer. Lifecycle management of business process variants. In J. vom Brocke and M. Rosemann, editors, *Handbook on Business Process Management 1, Introduction, Methods, and Information Systems, 2nd Ed.*, International Handbooks on Information Systems, pages 251–278. Springer, 2015.
30. M. Reichert, S. Rinderle, and P. Dadam. On the common support of workflow type and instance changes under correctness constraints. In *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE - OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003*, pages 407–425, 2003.
31. M. Reichert, S. Rinderle-Ma, and P. Dadam. Flexibility in process-aware information systems. *Transactions on Petri Nets and Other Models of Concurrency*, 2:115–135, 2009.
32. M. Reichert and B. Weber. *Enabling Flexibility in Process-Aware Information Systems - Challenges, Methods, Technologies*. Springer, 2012.
33. D. Riaño, R. Lenz, and M. Reichert, editors. *Knowledge Representation for Health Care - HEC 2016 International Joint Workshop, KR4HC/ProHealth 2016, Munich, Germany, September 2, 2016, Revised Selected Papers*, volume 10096 of *Lecture Notes in Computer Science*. Springer, 2017.
34. S. Rinderle, M. Reichert, and P. Dadam. Flexible support of team processes by adaptive workflow systems. *Distributed and Parallel Databases*, 16(1):91–116, 2004.
35. S. Rinderle, M. Reichert, and P. Dadam. On dealing with structural conflicts between process type and instance changes. In *Business Process Management: Second International Conference, BPM 2004, Potsdam, Germany, June 17-18, 2004. Proceedings*, pages 274–289, 2004.
36. M. L. Rosa, W. M. P. van der Aalst, M. Dumas, and A. H. M. ter Hofstede. Questionnaire-based variability modeling for system configuration. *Software and System Modeling*, 8(2):251–274, 2009.
37. E. Shalom, Y. Shahar, and E. Lunenfeld. An architecture for a continuous, user-driven, and data-driven application of clinical guidelines and its evaluation. *Journal of Biomedical Informatics*, 59:130–148, 2016.
38. W. M. P. van der Aalst, A. Dreiling, F. Gottschalk, M. Rosemann, and M. H. Jansen-Vullers. Configurable process models as a basis for reference modeling. In *Business Process Management Workshops, BPM 2005 International Workshops, BPI, BPD, ENEI, BPRM, WSCOBPM, BPS, Nancy, France, September 5, 2005, Revised Selected Papers*, pages 512–518, 2005.
39. W. M. P. van der Aalst, N. Lohmann, and M. L. Rosa. Ensuring correctness during process configuration via partner synthesis. *Information Systems*, 37(6):574–592, 2012.

40. W. M. P. van der Aalst, M. Pesic, and H. Schonenberg. Declarative workflows: Balancing between flexibility and support. *Computer Science - R&D*, 23(2):99–113, 2009.
41. B. Weber, M. Reichert, J. Mendling, and H. A. Reijers. Refactoring large process model repositories. *Computers in Industry*, 62(5):467–486, 2011.
42. B. Weber, M. Reichert, and S. Rinderle-Ma. Change patterns and change support features - enhancing flexibility in process-aware information systems. *Data Knowledge Engineering*, 66(3):438–466, 2008.
43. B. Weber, M. Reichert, and W. Wild. Case-base maintenance for CCBR-based process evolution. In *Advances in Case-Based Reasoning, 8th European Conference, ECCBR 2006, Fethiye, Turkey, September 4-7, 2006, Proceedings*, pages 106–120, 2006.
44. B. Weber, M. Reichert, W. Wild, and S. Rinderle. Balancing flexibility and security in adaptive process management systems. In *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, OTM Confederated International Conferences CoopIS, DOA, and ODBASE 2005, Agia Napa, Cyprus, October 31 - November 4, 2005, Proceedings, Part I*, pages 59–76, 2005.
45. B. Weber, S. Rinderle, W. Wild, and M. Reichert. CCBR-driven business process evolution. In *Case-Based Reasoning, Research and Development, 6th International Conference, on Case-Based Reasoning, ICCBR 2005, Chicago, IL, USA, August 23-26, 2005, Proceedings*, pages 610–624, 2005.
46. M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, 2007.