Computer Science - Research and Development manuscript No.
(will be inserted by the editor)

Barbara Weber · Shazia Sadiq · Manfred Reichert

Beyond Rigidity - Dynamic Process Lifecycle Support

A Survey on Dynamic Changes in Process-aware Information Systems

Submitted: date / Accepted: date

Abstract The economic success of an enterprise increasingly depends on its ability to react to changes in its environment in a quick and flexible way. To cope with emerging business trends, responsiveness to change is a significant competitive advantage. Similar to the lifecycle in conventional information systems development, studies on lifecycle support for business processes are often sweeping the issues of runtime change management under the banner of maintenance. However, the pervasiveness of dynamic changes in business processes warrants targeted attention. This paper presents a detailed review of challenges and techniques that exist for the lifecycle management of dynamic processes. For each of the lifecycle phases we discuss the needs and deliberate on various developments from both academia and industry.

Keywords Business Process Management · Dynamic Process · Business Process Lifecycle · Process-aware Information System

CR Subject Classification H.4.1 · D.2.2 · D.2.11

Barbara Weber
Department of Computer Science
Univ. of Innsbruck, Austria
Tel.: +43/512/507-6474
Fax: +43/512/507-9871
E-Mail: Barbara.Weber@uibk.ac.at

Shazia Sadiq
School of Inf. Technology and Electrical Eng.
The University of Queensland, Australia
Tel: +61 7 3365 3481
Fax: +61 7 3365 1999
E-Mail: shazia@itee.uq.edu.au

Manfred Reichert
Institute of Databases and Inf. Systems
Ulm University, Germany
Tel.: +49/731/50-24135
Fax: +49/731/50-24134
E-Mail: manfred.reichert@uni-ulm.de

1 Introduction

Historically speaking, business process support has been a major driver for enterprise information systems for a significant period of time. The overall goal is to overcome the drawbacks of functional over-specialization and lack of overall process control [9; 25; 38; 42]. Technology response to this business demand was met with a suite of technologies ranging from groupware and office automation, to workflow systems, and more recently to business process management technology. Just as database management systems provided a means of abstracting application logic from data logic, workflow management systems separate coordinative process logic from application logic. Every system generation has provided additional functionality through a variety of supporting tools. Although workflow management technology has delivered a great deal of productivity improvements, it has been mainly designed for the support of static (i.e., pre-defined) and repetitive business processes, which require a basic level of coordination between human performers and some application services.

More recently Business Process Management (BPM) has been used as broader term to reflect the fact that a business process may or may not involve human participants, and often crosses organizational boundaries. There is currently a wide spread interest on BPM technologies, especially in light of emerging paradigms surrounding web services and their application to dynamic process composition [42; 63]. In this context, the notion of PAIS (Process Aware Information System) provides a guiding framework to understand and deliberate on the above developments [12; 101]. In general, a PAIS architecture can be viewed as 4-tier system (cf. Fig. 1). As fundamental characteristic, a PAIS provides the means to separate process logic from application code. For this purpose, at *buildtime* the process logic has to be explicitly defined based on the elements provided by a process meta model (e.g., Workflow Nets [83] or WSM Nets [55]). At *runtime* the PAIS then orchestrates the processes according to the defined logic and coordinates process relevant applications and other resources. Examples of PAIS enabling technologies include workflow management systems like Staffware [12], WebSphere Process Server [22], ADEPT2 [49; 52], and YAWL [85] as well as case handling systems (e.g., FLOWer [87; 12; 43]).

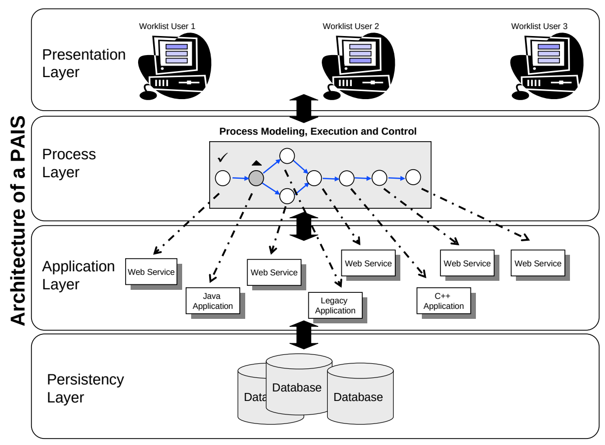


Abb. 1 Architecure of a PAIS

In spite of several success stories on the uptake of PAISs, and the growing process orientation of companies, so far, BPM and related technologies have not had the wide spread adoption that was expected. A major reason for this is the limited support of dynamic changes, which inhibits the ability of an organization to respond to business changes in an agile way [42; 84]. To deal with exceptions, uncertainty and evolving processes, it is widely recognized that a PAIS needs to provide runtime flexibility [97; 96; 78]. This can either be achieved through structural process changes and adaptations of the process state, or by supporting loosely specified process models, which can be refined during runtime according to predefined criteria and rules. To address this need for flexible and easily adaptable PAISs several competing paradigms for process change and process flexibility have been developed (e.g., adaptive processes [49; 1; 99], case handling [87], and declarative processes [46]).

This paper provides a detailed review of methods, tools and technologies provided to address flexibility issues in PAISs. Challenges, innovations and limitations of existing approaches are presented along the phases of the process lifecycle [94]. Further, we introduce, where necessary, additional lifecycle phases relevant to dynamic PAISs warranting flexibility in their operation. The aims of the paper can be summarized as follows:

- getting a clear picture of business process lifecycle management
- obtaining an appreciation and understanding of the unique set of challenges that dynamic processes face
- knowing the particular distinguishers of dynamic processes in all phases of the lifecycle
- getting a better understanding of how to ideally cope with process changes in information systems
- being aware of gaps in industry needs and existing solutions for lifecycle support of dynamic processes

To facilitate the discussion we first present a typical process lifecycle as it is promoted in both research and practice on BPM (cf. Fig. 2). The role of the different phases can be summarized as follows:

- **Design**. The design of business processes is primarily a management function driven by business objectives. The translation of such a high-level design into concrete (i.e., executable) models constitutes a pre-requisite for any PAIS.
- **Model**. Process modeling is a well studied field with formal (e.g., Petri Nets) and commercial (e.g., Business Process Modeling Notation) contributions. As with any modeling exercise, issues related to expressiveness and complexity have been widely debated. Section 3 discusses this phase in the context of dynamic processes.
- **Execute**. Flexible executability of process models is fundamental for the realization of dynamic processes. Issues relevant for this phase are discussed in Section 4. They include exception handling and treatment of unanticipated process changes.
- **Monitor**. Monitoring relates to traceability and post execution analysis, and is particularly important for dynamic processes. Through monitoring process improvements can be identified thus triggering subsequent process evolution. We discuss this phase in Section 5.

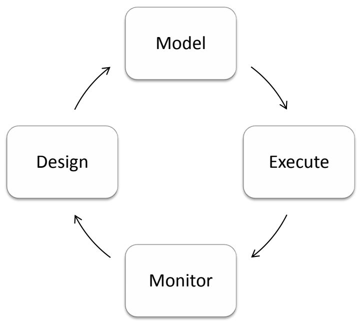


Abb. 2 Traditional Process Lifecycle

In this paper, our primary focus is on those phases of the process lifecycle managed through the technology infrastructure (i.e., we deal with executable processes), namely process modeling, execution and monitoring.

The paper is organized as follows. We first characterize the dynamic process and discuss the impact of its characteristics on the process lifecycle in Section 2. We then provide a detailed discussion on the methods, tools and technologies that are available for the phases of the lifecycle (i.e., modeling, execution and monitoring) and are affected by dynamic characteristics (cf. Sections 3, 4 and 5). Section 6 discusses a number of trade-offs that exist when designing a PAIS for the support of dynamic processes. Finally, Section 7 concludes with a summary.

2 Understanding the Dynamic Process

In order to discuss the characteristics of the *dynamic process*, we first introduce some basic terminology and a running example.

For each business process to be supported (e.g., handling a customer request or processing an insurance claim), a *process type* T represented by a *process schema* S has to be defined. For one particular process type several process schemes may exist, representing the different versions and the evolution of this process type over time. In the following discussion, we will use a typical notion of process schema, depicted as a directed graph, which comprises a set of *nodes* – representing process *activities* or *control connectors* (e.g., XOR-Split, AND-Join) – and a set of *control edges* (i.e., precedence relations) between them. Activities can either be *atomic* or *complex.* While an atomic activity is associated with an invokable application service, a complex activity contains a sub process or, more precisely, a reference to a sub process schema S'. This allows for the *hierarchical* decomposition of process schemes. In addition, a process schema comprises a set of *data elements* and a set of *data edges.* A data edge links an activity with a data element and represents a read or write data access of this activity.

Fig. 3a shows a simplified version of the control-flow perspective of a healthcare process representing a cruciate rupture treatment (in BPMN notation). The depicted process schema consists of ten activities and six control connectors: Activity `Patient Admission` is followed by activity `Anamnesis & Clinical Examination` in the flow of control, whereas activities `X-ray`, `MRT` and `Sonography` can be processed in parallel (i.e., in arbitrary order). Activities `Initial Treatment & Operation Planning` as well as `Operative Treatment` will be conditionally executed if the preceding medical examinations show that the patient is suffering from a cruciate rupture and non-operative treatment does not constitute a viable option. Based on the process schema depicted in Fig. 3a, at runtime new *process instances* can be created and executed (cf. Fig. 3b). Regarding process instance I_1 in Fig. 3b, for example, activities `Patient Admission`, `Anamnesis & Clinical Examination` and `X-ray` are completed, activity `Non Operative Therapy` is skipped, and `MRT` and `Sonography` are concurrently activated. Completion events related to the activities of a process instance are recorded in a *trace.* The traces for instances I_1 to I_4 are illustrated in Fig. 3b. Generally, as in the case of the above example, a large number of instances in different states may be present for a particular process schema.

Taking this example, we now consider selected characteristic scenarios for dynamic processes:

– The cruciate rupture treatment process depicted in Fig. 3a usually includes the following three diagnostic procedures: a magnetic resonance tomography (MRT), an X-ray, and a sonography. Assume now that this treatment process is performed for a patient with cardiac pacemaker. Then the MRT has to be skipped for this patient, i.e., the activity representing the MRT task has to be dynamically deleted from the corresponding process instance.
– A particular patient suffers from an effusion in his knee and the physician decides to conduct an additional puncture, i.e., a corresponding activity has to be dynamically added to a process instance.
– Different medical treatments exist, however, the exact ordering of the diagnostic and therapeutic procedures is decided during runtime. Process instances (representing the treatment process of a particular patient) are dynamically modelled or composed out of predefined activities.
– The MRT scan of a particular patient is blurred and thus has to be redone.
– Due to new legal regulations it becomes necessary to inform patients about alternative treatment methods before applying one of them. This requires the modification of the process schema (e.g., inserting activity `Inform Patient`). The law requires that this rule is applied to ongoing treatment processes as well (if still possible). As a consequence the schema of a process type has to be changed and respective changes have to be propagated to already running process instances of this type.

These scenarios identify a range of functions which are not easily manifested in a traditional process lifecycle. For example, instance-specific changes during runtime as needed for the patient with the cardiac pacemaker are only supported by few PAISs [97]. Also the support for loosely specified process models which can be refined during runtime does not constitute standard functionality of a contemporary PAIS. In fact these functions warrant specific extended functionality in the PAIS. Below we present an overarching taxonomy to define the problem space of dynamic process support (cf. Fig. 4).

We perceive dynamic processes to be characterized by three major requirements, namely support for flexibility, adaptation, and evolution. Each requirement has profound impact on various phases of the process lifecycle as illustrated in Fig. 4. Below we present a brief summary of each requirement to provide the basic rationale and motivation behind. This is followed by detailed discussions in the subsequent sections.

Flexibility represents the ability of the implemented process to execute on the basis of a loosely or partially specified model, which is fully specified at runtime and

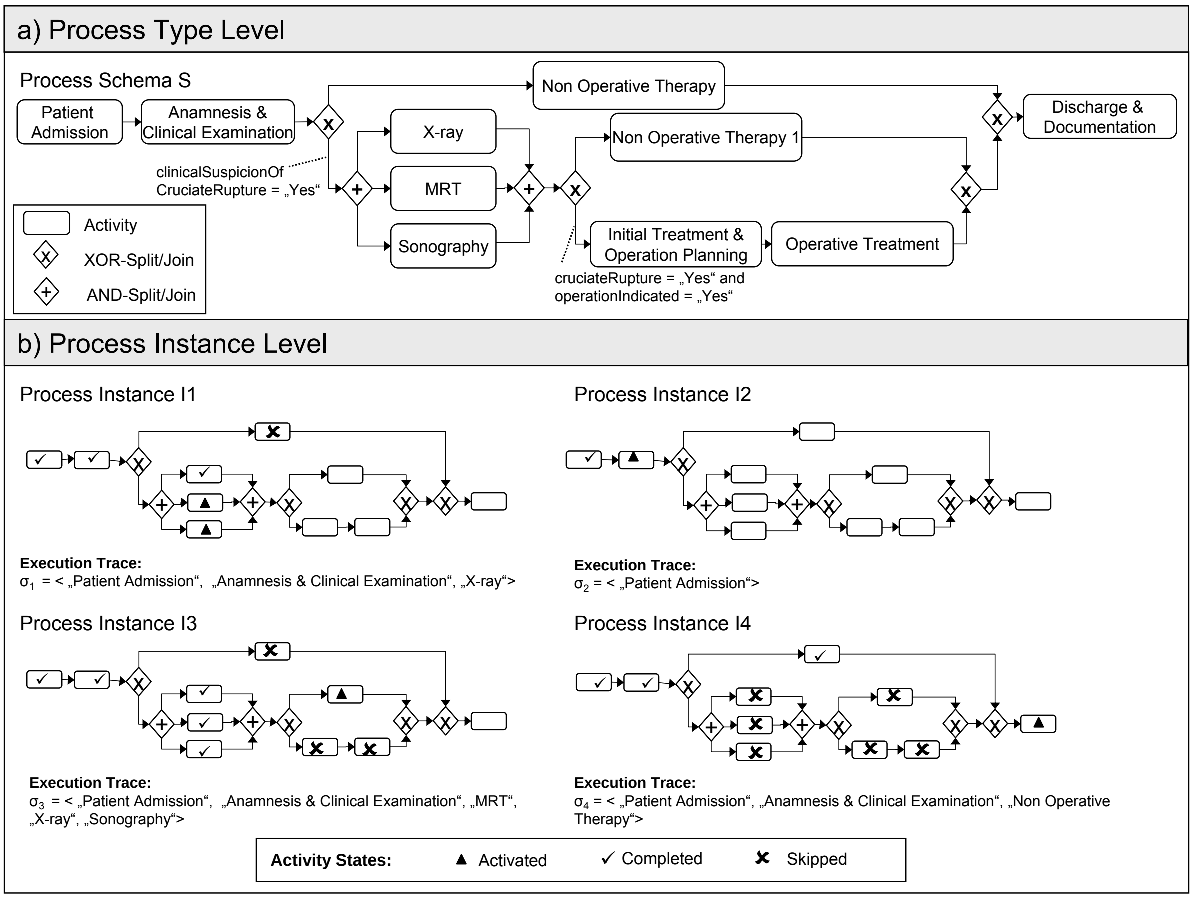

a) Process Type Level
Process Schema S
Patient Admission
Anamnesis & Clinical Examination
Non Operative Therapy
X-ray
MRT
Sonography
Non Operative Therapy 1
Initial Treatment & Operation Planning
Operative Treatment
Discharge & Documentation
clinicalSuspicionOf CruciateRupture = „Yes“
cruciateRupture = „Yes“ and operationIndicated = „Yes“
Activity
XOR-Split/Join
AND-Split/Join
b) Process Instance Level
Process Instance I1
Execution Trace:
σ1 = < „Patient Admission“, „Anamnesis & Clinical Examination“, „X-ray“>
Process Instance I2
Execution Trace:
σ2 = < „Patient Admission“>
Process Instance I3
Execution Trace:
σ3 = < „Patient Admission“, „Anamnesis & Clinical Examination“, „MRT“, „X-ray“, „Sonography“>
Process Instance I4
Execution Trace:
σ4 = < „Patient Admission“, „Anamnesis & Clinical Examination“, „Non Operative Therapy“>
Activity States: ▲ Activated ✓ Completed ✖ Skipped

Abb. 3 Core Concepts (Control-Flow Perspective)

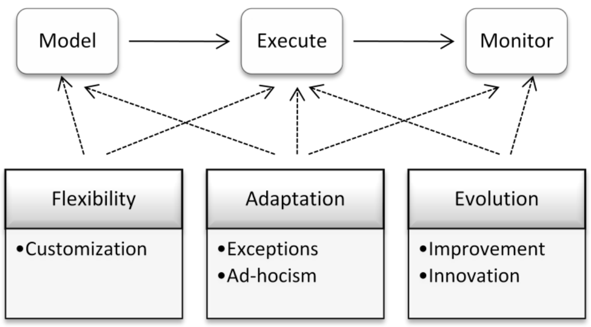


Abb. 4 Taxonomy for Characterizing Dynamic Processes

which may be unique to each instance. Due to the presence of a high degree of choices, not all of which can be anticipated and hence pre-defined, flexible processes need to be defined in a more relaxed or "flexible" manner. Thus, rather than enforcing control through a rigid or highly prescriptive model that attempts to capture every step and every option within the process, the model is defined in a way that allows individual instances to determine their own (unique) processes. Such an approach to modeling raises several challenges including the flexible configuration of process models at design time or their constraint-based definition during runtime.

Adaptation represents the ability of the implemented processes to cope with exceptional circumstances. On the one hand, this includes the handling of expected exceptions, which can be anticipated and thus captured in the process schema. On the other hand, this covers the handling of unanticipated exceptions, which usually are addressed through structural changes of single process instances (e.g., to add or delete activities).

Evolution represents the ability of the implemented process to change when the business process evolves. This evolution may be incremental as for process improvements, or radical as for process innovation or process re-engineering [44]. In any case, the assumption is that the processes have pre-defined models, and a change causes these models to be modified. The biggest problem here is the handling of running process instances, which were initiated based on the old model, but are required to comply with the new specification from now on. Since potentially thousands of active instances may be affected by a given process change the issue of compliance is rather critical. Traceability of changes and minining of dynamic processes are issues closely related to process evolution which also have to be considered.

3 Modeling Phase

The modeling phase typically constitutes the translation of business strategy into a process model in a given language. Before we discuss modeling issues for dynamic processes, it is important to establish a clear understanding of the process design phase, which in fact precedes the modeling phase as depicted in our lifecycle diagram (cf. Fig. 2). Process design is primarily a management function. During the era of process orientation, a number of approaches for the design of business processes have been proposed, including Six Sigma [45], Porter's value chain [47], Rummler's management theory [70], Capability Maturity Model [80], Process Improvement [21], Process Handbook [33], Reference Models [77], and Process Templates [76].

There exist significant contributions towards understanding the difficulties in translating the process design into executable models. In particular we refer to the work of Davenport[1] towards understanding and bridging the business-IT divide. It is argued that management pull and technology push need to be aligned, which does not naturally happen unless senior executives become intensively involved in process initiatives, and at the same time, BPM tools generate systems that are aligned with business strategies and hence demonstrate a clear outcome in terms of business value.

Another noteworthy contribution is the work on workflow patterns [86; 71; 72; 82]. Patterns allow a means of assessing the expressiveness of a given process modeling language which, in turn, provides an indication of its capability to align with process design. A pre-requisite to this analysis is that the strategy definition or process design can be related to a documented pattern. The complexity of establishing this relationship should not be underestimated due to the nature of process design documents that result from business strategic planning and articulation. That is, achieving a fit (lossless translation) is a task requiring great expertise.

Due to the high cost of process design and modeling, the modeling of dynamic processes is a lost cause to begin with. It is evident that the changes that a dynamic process will undergo, cannot always be anticipated and built into the original process model. No organization will be prepared to invest in an expensive process design and modeling initiative if it is known that the model will need to be changed almost immediately after its deployment. Therefore, a radically different approach is warranted for the modeling of dynamic processes.

[1] see www.tomdavenport.com for details

A key characteristic of dynamic processes is that they are knowledge intensive (e.g., [9; 25; 38]). Thus the process flow is governed by rich knowledge rather than a well-defined control flow. This knowledge is generally only tacitly available. That is, it cannot be found in corporate policy manuals or industry standards, but is derived from the experiences of domain experts (knowledge workers), or from external factors which may be unique for every case or process instance. Examples of such scenarios include healthcare, automotive engineering, customer relationship management, civil engineering, and many more. Note that translating business strategies into process models is already a difficult task at the presence of clear guidelines. Thus, the lack of explicit articulation makes process modeling extremely difficult for dynamic and knowledge intensive processes.

With the sketched background, we present below the major approaches for modeling dynamic processes.

3.1 Granularity Control

The granularity of a process activity has not been prescribed in any process modeling approach, and rightly so. Since an activity is a black box as far as the PAIS is concerned, it may actually comprise a number of sub-activities, whose existence or inter-dependencies are not disclosed to the process. As example consider activity `Anamnesis & Clinical Examination` in the process schema depicted in Fig. 3a. Obviously, this activity comprises several sub-activities (e.g., recording the medical history of the patient, physically examining the patient, documenting results of this examination) which are not explicitly defined in the process schema. This provides flexibility to the user regarding the order in which he performs these sub-activities. Generally, a simple option for modeling dynamic aspects of a process is to exclude them from process control, and hence have the flexibility in performing the sub-activities. Clearly, the application of this approach is rather limited. Whenever process control is required, granularity control will be meaningless.

3.2 Flexibility by Enumeration

Another way to achieve flexibility is to enumerate all possible execution paths in the process model. During runtime one specific execution path is then chosen. This kind of flexibility can be realized with any PAIS as it only requires alternative branching support. For dynamic processes this approach is only of limited use as it relies on the assumption that all possible execution paths can be predefined in advance.

3.3 Process Configuration

Typically, for a particular business process, different variants exist. Each of them constitutes an adjustment of a reference process to specific requirements. Commercial BPM tools do not adequately support the modeling and management of process variants. Either the process variants have to be specified in separate models or they are expressed in terms of conditional branches within the same process model (cf. Section 3.2). Both approaches, however, often lead to considerable model redundancies increasing maintenance efforts.

An important area related to process configuration is reference process modeling. Usually, a reference process has recommending character, covers a family of process models, and can be customized in different ways to meet specific needs. Configurable event-driven process chains (C-EPCs), for example, enable customization of reference process models [69; 24]. When modeling a reference process, activities (and decision nodes) in EPCs can be annotated to indicate whether or not they are mandatory. This information is considered when configuring the C-EPCs. A similar approach is presented in [89]. Here, the concepts for configuring a reference process model (i.e., to enable, hide or block configurable process elements) are transferred to executable (workflow) models.

Finally, Provop [20] provides an operational approach for managing process variants based on a master process model. In particular, process variants can be configured by applying a set of high-level change operations to a given master process. Provop supports context-aware process configuration, i.e., a process variant can be configured automatically by only applying those adjustments relevant in the given process context [19].

The approaches presented in [69], [89] and [24] allow for the configuration of process variants by removing activities from the reference process model (through hiding or blocking) and thus require the reference process model to contain all possible behavior. The Provop approach, in turn, suggests using change operations to adapt or extend the reference model in the desired way depending on the context. Thus, it is not required to capture all possible behavior in the reference process model.

3.4 Late Binding

The concept of late binding is well known in programming environments, and has been also used extensively in workflow systems for resource allocation (e.g., role based performer assignment) [101]. With Worklets a similar concept has been proposed for dynamic processes [1]. Worklets enable late binding of services or sub process fragments to process activities at runtime. Thus, at design time, the activity is merely modeled as placeholder. At runtime, an appropriate service is selected and bound to the process activity (cf. Fig. 5). Regarding our example, for instance, activity `X-ray` may be bound to an internal service of the hospital or to the service of an external provider in case this task shall be outsourced for the given case. Generally, the challenge is to ensure the structural (e.g., data flow compatibility) and semantic (contribution towards process goal) fit of the service.

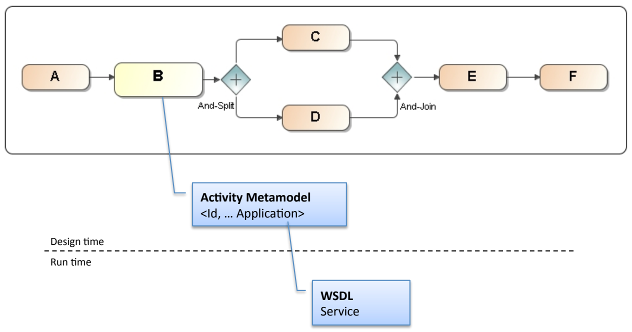


Abb. 5 Late Binding of Services to Activities

3.5 Late Modeling

Late modeling enables users to define parts or whole of the process at runtime. The model being defined at runtime may or may not be controlled by design guidelines or constraints. Complete lack of any constraints on the late modeling can defeat the purpose of a PAIS, whereas too many constraints may introduce a rigidity that compromises the dynamic process. Basically, there are four options for late modeling:

- **Option 1 (whole process, unconstrained)**. The whole process schema may be defined during runtime without need to consider any constraint.
- **Option 2 (parts of the process, unconstrained)**. Parts of the process are predefined, while parts can be flexibly defined during run-time.
- **Option 3 (parts of the process, constrained)**. Parts of the process may be well defined, whereas other parts of it can have a prohibitive number of execution options that can only be determined at runtime (under given constraints) based on tacit knowledge (e.g., a particular combination of case features or an expert opinion of a knowledge worker).
- **Option 4 (whole process, constrained).** This constitutes a special case of Option 3.

In [75] an approach for the late modeling of process fragments based on Option 3 (and Option 4 respectively) is proposed. This approach is illustrated by Fig. 6. A part of the process (indicated as activity B, and termed *Pocket of Flexibility*) is deemed to be of dynamic nature and is defined through a set of activities (`P`, `Q`, `R`) and a set of constraints defined on them (presented here in plain English for simplicity). At runtime, the *pocket of flexibility* is concretized for a given process instance based on tacit knowledge. Fig. 6 shows a particular design of the process, noting that it is also valid in terms of the prescribed constraints (i.e., activity `P` needs to be executed before `Q`). Regarding the healthcare domain, for example, a *pocket of flexibility* may represent a process step within a treatment process which covers clinical diagnostics (i.e., a set of diagnostic procedures and a set of constraints for their use). The concrete diagnostic procedures for a particular patient and their control flow are chosen during runtime. For a survey on modeling constraint networks as well as analyzing constraint satisfiability we refer to [29].

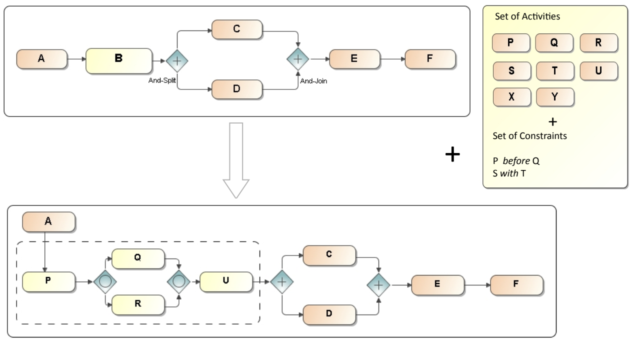


Abb. 6 Late Modeling

Similarly, DECLARE [46] provides support for the late composition of process fragments through a graphical constraint modeling language. Generally, when using DECLARE the whole process is defined in a declarative way (Option 4). However, DECLARE can also be used in combination with imperative modeling languages (e.g., YAWL) to realize Option 3 as well. In this particular scenario not the entire process model is defined in a declarative way, but only sub-processes. Like for the *Pocket of Flexibility* approach a process model is defined as a set of activities and a set of constraints (cf. Fig. 7a). During runtime process instances can be composed whereby any behavior not prohibited by any constraint is allowed. Fig. 7a depicts a declarative process model consisting of 6 activities and 2 constraints. Activities `A` and `B` are mutually exclusive, while there exists a response constraint between activities `C` and `F`, i.e., `F` must be executed eventually after `C`.

Existing challenges for late modeling include the specification and (efficient) verification of constraints as well as adequate end-user support for late modeling.

This section has discussed the specification and verification of constraints for late modeling. Issues related to runtime support are discussed in Section 4.3.

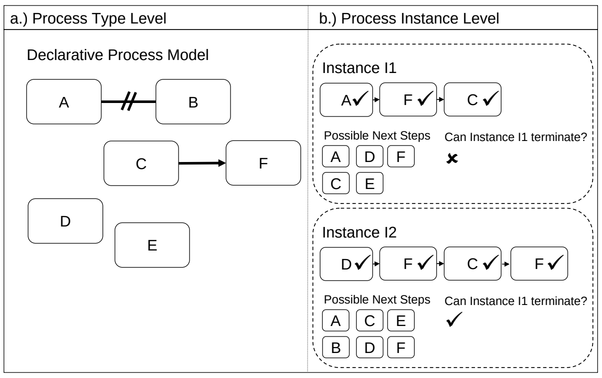


Abb. 7 Late Composition in DECLARE

4 Execution Phase

Runtime flexibility is one of the core challenges for any PAIS and requires the ability to deal with exceptions, uncertainty and evolving processes. While expected exceptions are usually considered during buildtime by specifying exception handlers [73], non-anticipated situations may require structural process adaptations [97]. In addition, to be able to efficiently deal with uncertainty, a PAIS should support loosely specified process models that can be refined during runtime. Section 4.1 discusses issues related to the handling of expected exceptions and Section 4.2 introduces structural process adaptations. Finally, Section 4.3 covers strategies for dealing with uncertainty.

4.1 Dealing with Expected Exceptions

The handling of expected exceptions has been widely discussed in literature and many different approaches have been proposed to effectively cope with this issue [8; 13; 32; 18; 41].

A comprehensive overview of ways to deal with exceptions is provided by [73]. Depending on the type of exception that is detected during process execution different exception handling strategies can be pursued. [73] represents such strategies as exception handling patterns. Each strategy describes (1) how the work item on which the exception is based should be handled, (2) how the process instance for which the exception is raised (and other related instances) shall be handled, and (3) what recovery actions are undertaken. In general, work items are handled by changing the state of a process instance (i.e., its behavior), but not its schema (i.e., its structure). For example, in the context of our running example, the MRT scan for a particular patient might be blurred, thus requiring the respective activity to be redone. In addition, state changes can also affect the current process instance or a related process instance (e.g., cancel process instance). Finally, it might be necessary to compensate or rollback the effects of the exception.

Exception handling often requires combined use of several exception handling patterns resulting in rather complex routines. The Exlet approach [2; 3] addresses this problem by allowing for the combination of different exception handling patterns to an exception handling process called Exlet. An Exlet is executed in parallel to the corresponding process instance and may be reused in other context when similar exceptions re-occur.

The ADEPT2 process management system provides comprehensive support for dealing with both anticipated and non-anticipated exceptions [48]. For example, ADEPT2 enables different kinds of forward and backward jumps in the flow of control while preserving data consistency of the corresponding process instance [50].

4.2 Dealing with Unanticipated Changes

While exception handling patterns are well suited for dealing with expected exceptions, non-anticipated situations often require structural adaptations of single process instances during runtime [97]. Such *ad-hoc changes* lead to an adapted process instance schema [49]. In particular, their effects are instance-specific and must not affect any other process instance. As example consider our cruciate rupture treatment process, which usually includes activities `magnetic resonance tomography (MRT)`, `X-ray`, and `sonography`. As aforementioned, for a particular patient the MRT may have to be skipped as he has a cardiac pacemaker (cf. Fig. 8). The corresponding activity deletion is instance-specific and must therefore not affect the treatment process of any other patient.

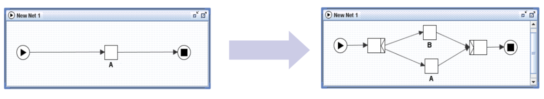
In the following we discuss fundamental requirements related to ad-hoc process changes at the instance level: conducting ad-hoc changes at a high level of abstraction, ensuring correctness of modified process instances, access control for ad-hoc changes, user assistance and change reuse, and handling of concurrent changes.

4.2.1 Support for Changes at High-level of Abstraction

Two different options exist for realizing structural adaptations of a process schema [97]. On the one hand, structural adaptations can be realized based on a set of *change primitives* like `add node`, `remove node`, `add edge`, `remove edge`, and `move edge`. Following this approach, the realization of a particular adaptation (e.g., to delete an activity or to add a new one) usually requires the application of multiple change primitives. Specifying structural adaptations at this low level of abstraction, however, is a complex and error-prone task. Generally, when applying a single change primitive (e.g., WASA2 [99], CAKE2 [37] or InConcert [102]), soundness of the resulting process schema (e.g., absence of deadlocks) cannot be guaranteed. Therefore, for more complex process meta models it is not possible to associate formal pre-/post-conditions with the application of single primitives. Instead, schema correctness has to be explicitly checked after applying the respective set of primitives.

On the other hand, structural adaptations can be based on *change patterns*, i.e., high-level change operations (e.g., to insert a process fragment between two sets of nodes), which abstract from the concrete schema transformations to be conducted. Instead of specifying a set of change primitives the user applies one or more high-level change operations to realize the desired process schema adaptation. Approaches following this direction often associate pre- and post-conditions with the high-level operations, which enables the PAIS to guarantee soundness when applying corresponding change operations [49; 7]. Note that soundness becomes a fundamental issue when changes are applied by end users or even more challenging by automated software agents [4; 14; 10; 41].

Fig. 9 illustrates the benefit of using change patterns instead of change primitives. The original process schema on the left side consists of a single activity `A`. Assume now that a process change shall be accomplished inserting activity `B` in parallel to `A`. On the one hand this change can be accomplished by using one high-level change operation `parallelInsert (S, B, A)` which adds activity `B` parallel to `A`. Applying this change pattern the user just has to specify a couple of parameters. On the other hand, change primitives can be used to realize the desired adaptation. In the given example, the transformation of the original process schema into the new schema version requires nine change primitives (cf. Fig. 9). Using the high-level operation `parallelInsert` instead, from the perspective of the user, eight operations can be saved.



Insert B parallel to A

Change Patterns

1 High-Level Change Operation

```
parallelInsert (S, B, A)
```

Change Primitives

9 Change Primitives

```
Add Node (B)
Add Node (AND-Split)
Add Node (AND-JOIN)
Move Edge ((Start, A), (Start, AND-Split))
Move Edge ((End, A), (AND-Join, End)
Add Edge (AND-Split, A)
Add Edge (AND-Split, B)
Add Edge (A, AND-Join)
Add Edge (B, AND-Join)
```

Abb. 9 Change Patterns vs. Change Primitives

[96; 97] provide a catalog of 14 change patterns (cf. Fig. 12) and an evaluation of selected PAISs along them. For a description of the formal semantics of process change patterns we refer to [67].

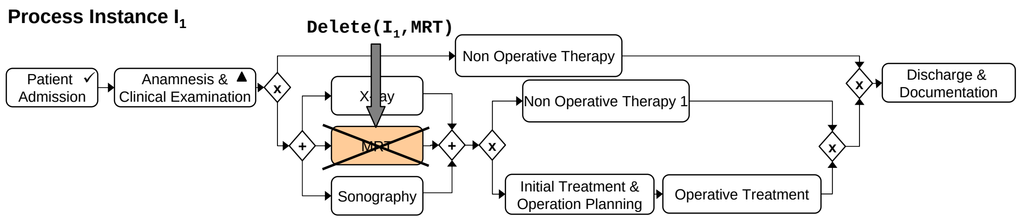


Abb. 8 Example for an Instance-specific Change

Adaptation Patterns	
AP1: **Insert Process Fragment**	AP8: **Embed Process Fragment in Loop**
AP2: **Delete Process Fragment**	AP9: **Parallelize Activities**
AP3: **Move Process Fragment**	AP10: **Embed Process Fragment in Conditional Branch**
AP4: **Replace Process Fragment**	AP11: **Add Control Dependency**
AP5: **Swap Process Fragment**	AP12 **Remove Control Dependency**
AP6: **Extract Sub Process**	AP13: **Update Condition**
AP7: **Inline Sub Process**	AP14: **Copy Process Fragment**

Abb. 10 Catalogue of Adaptation Patterns

4.2.2 Correctness of Ad-hoc Changes

An essential component of any change framework must be to ensure correctness and consistency of process instances when dynamically adapting them. First, structural and behavioral soundness of the modified process schema should be guaranteed independent from whether or not the change is applied at the instance level. Furthermore, when performing structural schema changes at the instance level, this must not lead to inconsistent or erroneous process states afterwards. Therefore, a correctness criterion is needed to decide whether or not a given process instance is *compliant* with a modified process schema [7; 99; 74; 59]. A detailed overview of correctness issues emerging in connection with ad-hoc process changes can be found in [57; 68]. Finally, a recently conducted evaluation of flexible PAISs shows that correctness issues in the context of ad-hoc changes are often ignored [97].

4.2.3 Access Control for Changes

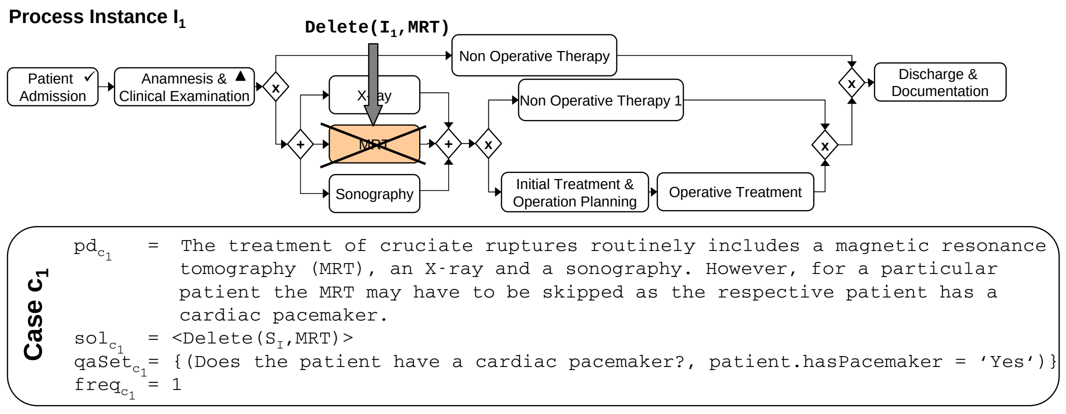
The support of runtime changes leads to increased process flexibility. This imposes several security issues as the PAIS becomes more vulnerable to misuse [93; 11]. Therefore, the application of changes at the process type as well as the process instance level should be restricted to authorized users. An example of an access control framework existing in this context is SecServ [93], which is a security service tailored towards the specific needs of adaptive PAISs enabling runtime flexibility. SecServ allows for both *user dependent* and *process type dependent* access rights. While the former restrict access to authorized users in order to avoid misuse (e.g., only users with role *physician* are authorized to insert the `X-ray` activity), the latter access rights are applied to only allow for change commands useful within a particular context (e.g., activity *vacation request* must not be inserted in medical treatment processes). SevServ supports both the specification and validation of static and dynamic constraints (i.e., constraints considering runtime data).

4.2.4 User Assistance and Change Reuse

For the practical applicability of a flexible PAIS end-user support is crucial. Although the use of change patterns ensures that no errors or inconsistencies are introduced, a flexible PAIS demands for more advanced user support. In the context of unplanned instance changes "similar" deviations (i.e., combinations of one or more change pattern) often occur more than once. Think of, for example, the clinical domain where patients with similar problems may have to be treated over and over again. Generally, the from scratch definition of changes requires significant user experience. To improve this situation several proposals for facilitating ad-hoc changes through change reuse have been made [61; 31; 35; 36; 98; 92; 95]. For example, [98] presents a case-based reasoning approach for annotating changes with contextual information (e.g., about the reasons for the deviation). Corresponding annotations are then memorized together with the changes and can be retrieved later in similar problem situations. Fig. 11 illustrates such a memorized case along our running healthcare example.

4.2.5 Controlling Concurrent Changes

Any PAIS supporting instance-specific adaptations should be able to cope with *concurrent* ad-hoc changes. When two users want to apply different ad-hoc changes to a particular process instance at the same time, the PAIS has to ensure that no inconsistencies or errors occur. In addition, it has to be guaranteed that no inconsistent states are introduced, when process instances are changed, but their execution proceeds. The easiest way to avoid such problems is to prohibit concurrent changes. WASA2 [100], for example, locks the entire process instance when an instance-specific change is performed.



```
pd_c1     =  The treatment of cruciate ruptures routinely includes a magnetic resonance
             tomography (MRT), an X-ray and a sonography. However, for a particular
             patient the MRT may have to be skipped as the respective patient has a
             cardiac pacemaker.
sol_c1    = <Delete(S_I,MRT)>
qaSet_c1  = {(Does the patient have a cardiac pacemaker?, patient.hasPacemaker = 'Yes')}
freq_c1   = 1
```

Abb. 11 Using Context Information to Foster Change Reuse [98]

FLOWer [87], in turn, avoids change conflicts by prohibiting users to simultaneously change the same case (i.e, process instance). As opposed to these approaches, ADEPT2 [52] and CAKE2 [35] provide support for concurrent changes. While ADEPT2 supports concurrent ad-hoc changes of a particular process instance based on optimistic techniques, locking is used in CAKE2. The breakpoint mechanism provided by CAKE2 only suspends those parts of a process instance which have to be changed. Parallel branches, not affected by the change, may proceed with their execution.

4.3 Dealing with Uncertainty

In addition to the handling of expected and unexpected exceptions a flexible PAIS must be able to deal with uncertainty. There exist several proposals on how to deal with this challenge [1; 75; 46; 87]. Common to all of them is the idea to cope with uncertainty by deferring decisions regarding the exact flow of control to runtime. Instead of requiring a process schema to be fully specified prior to execution, parts of the schema remain unspecified. Examples for such techniques are *Late Binding*, *Late Modeling* and *Late Composition of Process Fragments* (see also Sections 3.4 and 3.5) as well as *Multi-Instance Activities*. Finally, data-driven processes have been proposed in this context as well [87; 56; 39; 40].

Late Binding allows deferring the selection of the implementation of a particular process activity (e.g., a Web service or a Java program) to runtime. The concrete implementation is selected either based on predefined rules or on user decisions.

Late Modeling offers more freedom and allows for modeling selected parts of the process schema at runtime. Prior to execution only a placeholder activity has to be provided, its implementation is modeled during runtime. The modeling of the placeholder activity needs to be completed before its execution may start. For example, the process variant depicted in Fig. 6 replaces the placeholder activity `B` with a process fragment comprising activities `P`, `Q`, `R` and `U`. The depicted variant is valid as it obeys all execution constraints imposed by the process model. A particular variant of late modeling is provided by *Late Composition*, which enables the on-the-fly composition of process fragments from the process repository (e.g., by dynamically introducing control dependencies between a predefined set of activities and process fragments, respectively). There is no predefined schema, but the process instance is created in an ad-hoc way by selecting from the available activities in the repository and obeying the predefined constraints. Fig. 7b shows two process instances which have been created based on the declarative process model depicted in Fig. 7a. Considering the partial trace of instance I_1 possible activities to be executed next are `A`, `C`, `D`, `E` and `F`. Activity `B` cannot be executed for I_1 as it is mutually exclusive with `A`. In its current state I_1 cannot be terminated as the response constraint between activities `C` and `F` is violated. By contrast, process instance I_2 has an execution state for which termination is possible.

Moreover, *Multi-Instance Activity* allows for deferring the decision on how often a specific activity shall be executed to runtime, while the activity itself needs to be predefined at design time [86]. [56] discusses how the multi-instance pattern can be used to deal with certain kinds of exceptions during process execution.

Data-driven processes, as realized by [87; 39; 40; 53; 23], offer promising perspectives with respect to flexible process execution and adaptation.

The case-handling system FLOWer [87], for example, does not predefine the exact flow of control, but orchestrates the execution of activities based on the data assigned to a case (i.e., process instance). Thereby, different kinds of data objects are distinguished (cf. Fig. 12). Mandatory and restricted data objects are explicitly linked to one or more activities. If a data object is mandatory for an activity, a value has to be assigned to it before the activity can be completed (e.g., Data Object 5 in Fig. 12). If a data object is restricted for an activity, this activity needs to be active in order to assign a value to the data object. Free data objects, in turn, are not explicitly associated with a particular activity and can be changed at any point in time during case execution (e.g., Data Object 3 in Fig. 12). This, in turn, provides additional flexibility during runtime. In [16] a qualitative comparison of case-handling an flexible workflow technology is given, while [43] compares both technologies along a controlled experiment.

Another data-driven approach is provided by COREPRO [39; 40], which enables data-driven creation and adaptation of large process structures during runtime. For this purpose COREPRO establishes a strong linkage between (product) data structures and process structures. In particular, it translates (dynamic) changes of a currently processed data structure into corresponding adaptations of the related process structure under execution. COREPRO uses comprehensive consistency criteria in order to ensure that dynamic adaptations of a process structure lead to a correct process structure again. In summary, COREPRO addresses the full life cycle of data-driven, dynamic process structures.

Change reuse is not only relevant for structural process adaptations, but also for loosely specified process models. The more flexibility is provided, the bigger the need for user support becomes. In the context of late binding, [1] suggests incrementally evolving selection rules. To better support late modeling, [30] fosters reuse by making past process instances available to the user through a search interface and by saving frequently reoccurring instances as templates. Alternatively, in the context of declarative processes, [79] proposes the usage of log-based recommendations to support users in selecting among several different options the one which meets the user's goals best. Finally, in [90] a recommendation service for product data models is presented. Like in [79] recommendations are used for assisting users in selecting the step meeting performance goals of the process best.

5 Monitoring and Diagnosing Phase

The monitoring and diagnosing phase is particularly important for dynamic processes. It comprises support for both the real-time monitoring of the state and structure of dynamically evolving process instances and the post-execution analysis of completed ones.

Through monitoring authorized users (e.g., process administrators) can get a quick overview of the execution status and structure of a particular process instance (e.g., whether or not instance execution is delayed) or a collection of process instances (e.g., using a business dashboard which provides an abstract visualization of the status of these processes). Process diagnosis, in turn, focuses on the comprehensive analysis of a collection of process instances in order to identify potential process improvements. The latter can be accomplished, for example, by evolving the corresponding process schema accordingly. In the context of long running processes, this additionally necessitates support for the controlled migration of running process instances to the new schema version.

Section 5.1 discusses aspects related to the traceability of dynamic processes and Section 5.2 presents selected approaches for diagnosing and mining dynamic processes. In Section 5.3 we show how identified process optimizations can be realized in an adaptive PAIS. Finally, Section 5.4 discusses further relevant issues.

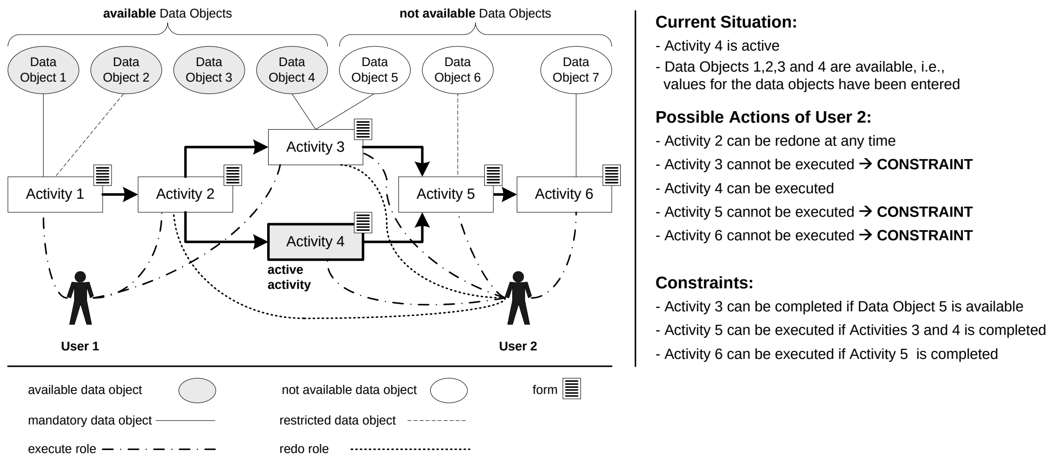
5.1 Traceability of Changes and Monitoring of Dynamic Processes

To ensure traceability of a dynamic process the PAIS must be able to restore both its execution schema and its execution state at any point in time. Generally, for dynamic processes it is therefore not sufficient to only log normal execution events (e.g., related to the start or completion of activities) in corresponding traces, but it additionally becomes necessary to store information about applied process adaptations in change logs [62; 64]. Further, it should be possible to correlate change log entries with semantical information (e.g., about the reasons and the context of the applied instance changes [98]). The latter is important for analyzing ad-hoc deviations and for learning from these changes.

By utilizing execution logs and correlating them with application-specific data, for example, the PAIS may provide sophisticated components for real-time monitoring as well as for process performance management (e.g., evaluating a set of key performance indicators). The latter enables comprehensive analyses of the execution data of process instance collections and requires sophisticated support for business process visualization [5; 6].

5.2 Diagnosing and Mining of Dynamic Processes

A PAIS supporting dynamic processes stores information about ad-hoc deviations in change logs. Obviously, corresponding log information provides promising perspectives with respect to the diagnosis and analysis of dynamic processes. In [15; 17] two change mining techniques are presented: multiphase change mining and mining of changes processes with regions. Both approaches do not only analyze the execution logs of the operational processes, but also consider the changes applied at the process instance level. The change processes discovered through process mining provide an aggregated overview of all changes that happened so far. Using process mining as analysis tool, [17] shows how better support can be provided for dynamic processes by understanding when and why process changes become necessary.



Current Situation:
- Activity 4 is active
- Data Objects 1,2,3 and 4 are available, i.e., values for the data objects have been entered

Possible Actions of User 2:
- Activity 2 can be redone at any time
- Activity 3 cannot be executed → **CONSTRAINT**
- Activity 4 can be executed
- Activity 5 cannot be executed → **CONSTRAINT**
- Activity 6 cannot be executed → **CONSTRAINT**

Constraints:
- Activity 3 can be completed if Data Object 5 is available
- Activity 5 can be executed if Activities 3 and 4 is completed
- Activity 6 can be executed if Activity 5 is completed

Abb. 12 Data Driven Case Handling (adapted from [43])

In the MinAdept project [27] even more advanced algorithms for mining a collection of process (instance) variants have been developed. For example, [26] provides a cluster-based technique that fosters learning from past process changes by mining the corresponding collection of configured process variants. As result we obtain a generic process model for which the average distance between this model and the mined process variants (i.e., number of high-level change operations needed to transform one model into the other [28]) becomes minimal. By adopting this generic model as reference model in the PAIS, need for future process adaptations at the instance level decreases.

5.3 Process Schema Evolution

In order to deal with the evolving nature of dynamic processes (e.g., to cope with changes of legal regulations or to implement process optimizations), a PAIS must support process schema changes at the type level (in the following also denoted as *schema evolution*). Such a process schema evolution may also require the propagation of the changes to ongoing process instances, particularly if these instances are long-running [51]. For example, let us assume that in a patient treatment process, due to new legal requirements, patients have to be educated about potential risks before a surgery takes place. Let us further assume that this change is also relevant for patients for which the treatment has already been started. In such a scenario, stopping all ongoing treatments, aborting them, and re-starting them is not a viable option. As a large number of treatment processes might be concurrently running, applying this change manually to all ongoing treatment processes is also not a feasible option. Instead, efficient PAIS support is required to add this new activity to all patient treatments for which this is still feasible (e.g., if the surgery has not yet started).

Generally, to support changes at the process type level, version control for process schemes is needed. In case of long-running processes, as motivated, the controlled migration of already running instances from the old to the new process schema version should be possible when conducting a schema change at the type level [59].

If no version control is provided, either the designer will have to manually create a copy of the process schema to be changed or the original schema will be overwritten. In the latter case, running instances can either be withdrawn from the runtime environment or they remain associated with the modified schema. Depending on current instance execution state and on how changes are propagated to instances progressed too far, missing version control can lead to inconsistent states and – worst case – to deadlocks or other severe runtime errors.

By contrast, if a PAIS provides explicit version control already running process instances may remain associated with the old schema version, while new instances can be created based on the new schema version. This approach leads to the co-existence of process instances belonging to different schema versions.

Furthermore, for an already running process instance a controlled migration to the new process schema version will be possible, if the instance in its current execution state is compliant with the new schema version (see Fig. 13 for example). In the latter context the PAIS should be also able to deal with "concurrent" changes at the process type and the process instance level (see [51; 60; 58] for detailed discussions on this topic). In this context well elaborated correctness criteria are needed to not needlessly exclude process instances from being migrated to the new schema version [68].

As shown in the DYCHOR project [63] things become even more complicated when changing the schema of a process choreography, i.e., the interaction protocol representing the coordination of multiple processes.

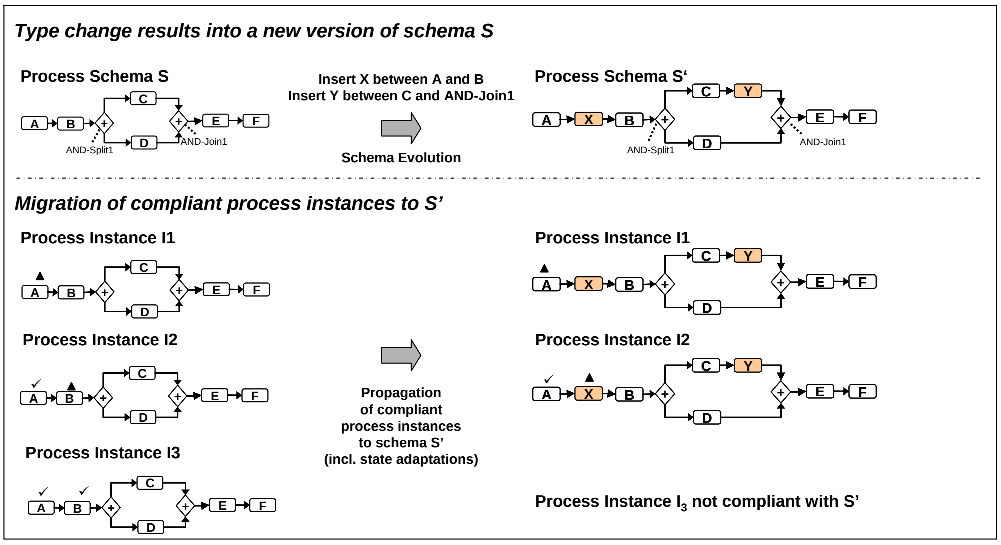

Type change results into a new version of schema S
Process Schema S
Insert X between A and B
Insert Y between C and AND-Join1
Schema Evolution
Process Schema S‘
AND-Split1
AND-Join1
Migration of compliant process instances to S’
Process Instance I1
Process Instance I2
Process Instance I3
Propagation of compliant process instances to schema S’ (incl. state adaptations)
Process Instance I3 not compliant with S’

Abb. 13 Migrating compliant process instances to a new process schema version.

5.4 Further Issues

With the increasing adoption of PAISs large process repositories will emerge over time [54]. The evolution of process models bears the risk that redundancies are introduced leading to unnecessary complexity [91]. When no continuous efforts are undertaken to keep process models simple, not only changes will become increasingly complex and time-consuming, but errors become more probable as well (as a recently conducted study by [34] revealed). In this context [91] proposes 11 behavior-preserving refactoring techniques, which are especially suited for business process models (cf. Fig. 14). These refactorings are applicable to (hierarchically structured) process models as well as to process model variants derived from a generic process schema. Respective refactoring techniques allow process designers to improve the design of process models and to remove redundancies introduced as the result of model adaptations.

Catalogue of Process Model Refactorings	
Refactorings for Process Model Trees	RF7: **Re-label Collection**
RF1: **Rename Activity**	RF8: **Remove Redundancies**
RF2: **Rename Process Schema**	**Refactorings for Process Variants**
RF3: **Substitute Process Fragment**	RF9: **Generalize Variant Change**
RF4: **Extract Process Fragment**	**Refactorings for Model Evolution**
RF5: **Replace Process Fragment by Reference**	RF10: **Remove Unused Branches**
RF6: **Inline Process Fragment**	RF11: **Pull Up Instance Change**

Abb. 14 Catalogue of Process Model Refactorings

6 Discussion

When designing PAISs, which provide support for dynamic processes, several trade-offs exist. The most important ones are discussed in this section.

Trade-off between degree of control in imperative and in declarative approaches. Imperative approaches allow to precisely prescribe the control as well as the data flow of the processes to be implemented. This, in turn, provides the basis for comprehensive correctness checks, seamless integration of application services (with input and output parameters), and many other useful PAIS functions, but at the price of less a-priori-flexibility. However, as shown in the previous sections, there are imperative approaches which provide a high degree of runtime flexibility by allowing authorized users to deviate from the prescribed flow of control. As opposed to imperative approaches, declarative modeling approaches only describe what to do, but do not prescribe in detail how to do things. While this provides more in-build-flexibility to users, with increasing complexity of the processes it will decrease ease-of-use and aggravate the implementation of more advanced PAIS functions (e.g., concerning application integration). Obviously, both paradigms have their right to exist and are certainly not able to completely replace each other.

Trade-off between expressiveness and flexibility in imperative approaches. Obviously, there exists a trade-off between expressiveness of a process meta model and the provided support for dynamic structural adaptations. For example, ADEPT2 has been designed with the goal to enable the latter [49]. To allow for an efficient implementation of adaptation patterns, well elaborated restrictions on the process meta model have been made. Similar restrictions in terms of expressiveness hold for other approaches supporting structural adaptations (e.g., CAKE2 [35] and WASA2 [100]). On the other hand, YAWL is a reference implementation for process patterns and therefore allows for a high degree of expressiveness [85]. Structural adaptations have not yet been addressed in YAWL and their implementation would be more

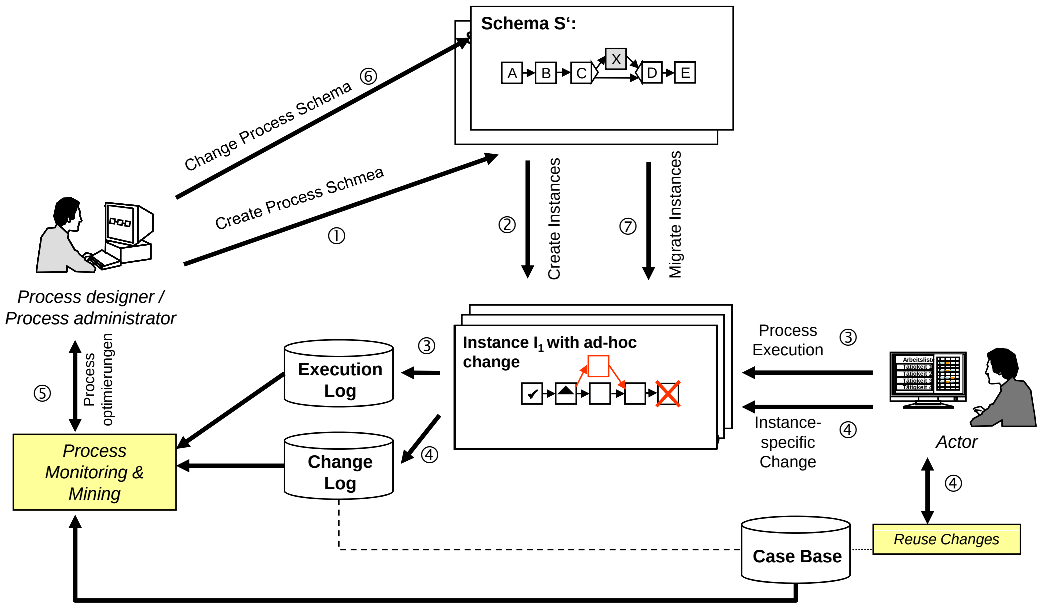


Abb. 15 Revised Process lifecycle

difficult due to the higher expressiveness. However, the integration of Worklets/Exlets with YAWL has shown that late binding can be easily realized for expressive process meta models as well [1; 2]. Furthermore, the declarative process management tool DECLARE [46] has been integrated as a service for YAWL, which allows for the runtime composition of sub process fragments in a declarative way.

Trade-off between flexibility and support in declarative approaches. It seems that the trade-off between expressiveness and flexibility does not exist to the same degree for declarative systems. Corresponding systems can be characterized as both expressive and flexible. However, compared to imperative approaches the potential for process automation is limited. As declarative process models allow for any behavior not prohibited by constraints, process models tend to provide several alternative options how to proceed with a process instance. Whenever several options exist user interaction is needed to select the most appropriate one. With increasing size and complexity of process models the task of selecting the best alternative path to proceed, becomes increasingly difficult for end users. Consequently, advanced user support is needed to guide end users through the process.

7 Summary and Outlook

To reflect the specific challenges for the lifecycle management of dynamic processes the simple life cycle model depicted in Fig. 2 needs to be complemented. Fig. 15 illustrates the revised lifecycle model. At buildtime an initial representation of a business process is created either by explicitly modeling the process (based on analysis results) or by discovering process models through the mining of execution logs [88] (1). At runtime new process instances can be derived from the predefined process schema (2). In general, process instances are executed according to the process type schema they were derived from, and (non-automated) activities are assigned to process participants to perform the respective activities (3). However, when exceptional situations occur during runtime, process participants may deviate from the predefined schema (4). While execution logs record information about the start and completion of activities as well as their ordering, process changes are recorded in change logs (5). The analysis of respective logs by a process engineer and process intelligence tools respectively allows discovering malfunctions or bottlenecks [81; 88]. This information often results in an evolution of the process schema (6). If desired, changes can be propagated to running process instance (7).

It it worth mentioning that work on dynamic processes has to be complemented by approaches providing similar flexibility for process aspects other than control and data flow. For example, in the CEOSIS project [65; 66], (dynamic) changes of organizational models as well as access constraints are studied in-depth.

Literatur

1. Adams M, ter Hofstede A, Edmond D, van der Aalst W (2006) Worklets: A service-oriented implementation of dynamic flexibility in workflows. In: Proc. Coopis'06, LNCS 4275, pp 291–308

2. Adams M, ter Hofstede A, Edmond D, van der Aalst W (2007) Dynamic and extensible exception handling for workflows. Tech. Rep. BPM-07-03, BPMcenter.org
3. Adams M, ter Hofstede A, van der Aalst W, Edmond D (2007) Dynamic, extensible and context-aware exception handling for workflows. In: Proc. CoopIS'07, LNCS 4803
4. Bassil S, Keller R, Kropf P (2004) A workflow-oriented system architecture for the management of container transportation. In: Proc. BPM'04, LNCS 3080, pp 116–131
5. Bobrik R, Bauer T, Reichert M (2006) Proviado personalized and configurable visualizations of business processes. In: Proc. EC-WEB'06, LNCS 4082, pp 61–71
6. Bobrik R, Reichert M, Bauer T (2007) View-based process visualization. In: Proc. BPM'07, LNCS 4714, pp 88–95
7. Casati F (1998) Models, semantics, and formal methods for the design of workflows and their exceptions. PhD thesis, University of Milano
8. Casati F, Fugini MG, Mirbel I (1999) An environment for designing exceptions in workflows. Inf Syst 24(3):255–273
9. Dadam P, Reichert M, Kuhn K (2000) Clinical workflows – the killer application for process-oriented information systems? In: Proc. BIS'00, Poznan, Poland, pp 36–59
10. de Leoni M, Mecella M, de Giacomo G (2007) Highly dynamic adaptation in process management systems through execution monitoring. In: Proc. BPM'07, LNCS 4714, pp 182–197
11. Domingos D, Rito-Silva A, Veiga P (2003) Authorization and access control in adaptive workflows. In: Proc. ESORICS'03, pp 23–38
12. Dumas M, ter Hofstede A, van der Aalst W (eds) (2005) Process Aware Information Systems. Wiley Publ.
13. Eder J, Liebhart W (1996) Workflow recovery. In: CoopIS'96, pp 124–134
14. Golani M, Gal A (2006) Optimizing exception handling in workflows using process restructuring. In: Proc. BPM'06, LNCS 4102, pp 407–413
15. Günther C, Rinderle S, Reichert M, van der Aalst W (2006) Change mining in adaptive process management systems. In: Proc. CoopIS'06, LNCS 4275, pp 309–326
16. Günther C, Reichert M, van der Aalst W (2008) Supporting flexible processes with adaptive workflow and case handling. In: Proc. WETICE'08
17. Günther C, Rinderle-Ma S, Reichert M, van der Aalst W, Recker J (2008) Using process mining to learn from process changes in evolutionary systems. Int'l Journal of Business Process Integration and Management, Special Issue on Business Process Flexibility 3(1):61–78
18. Hagen C, Alonso G (2000) Exception handling in workflow management systems. IEEE Transactions on Software Engineering 26(10):943–958
19. Hallerbach A, Bauer T, Reichert M (2008) Context-based configuration of process variants. In: Prof. TCoB'08, pp 31–40
20. Hallerbach A, Bauer T, Reichert M (2008) Managing process variants in the process lifecycle. In: Proc. ICEIS'08, Barcelona, pp 154–161
21. Harmon P (2003) Business Process Change - A Manager's guide to Improving, redesigning and Automating Processes. Morgan Kaufmann
22. Kloppmann M, Knig D, Leymann F, Pfau G, Roller D (2008) Business process choreography in WebSphere. combining the power of BPEL and J2EE. IBM Systems Journal 43(2):270–296
23. Küster J, Ryndina K, Gall H (2007) Generation of business process models for object life cycle compliance. In: BPM'07, LNCS 4714, pp 165–181
24. la Rosa M, Lux J, Seidel S, Dumas M, ter Hofstede A (2007) Questionnaire-driven configuration of reference process models. In: Proc. CAiSE'07, LNCS 4495, pp 424–438
25. Lenz R, Reichert M (2007) IT support for healthcare processes - premises, challenges, perspectives. Data and Knowledge Engineering 61(1):39–58
26. Li C, , Reichert M, Wombacher A (2008) Discovering reference process models by mining process variants. In: Proc. ICWS'08, Beijing, pp 45–53
27. Li C, , Reichert M, Wombacher A (2008) Mining process variants: Goals and issues. In: Proc. SCC'08, Honolulu, Hawaii, pp 573–576
28. Li C, Reichert M, Wombacher A (2008) On measuring process model similarity based on high-level change operations. In: Proc. ER'08, Barcelona, LNCS 5231, pp 248–264
29. Lu R (2008) Constraint based flexible business process management. PhD thesis, School of Information Technology and Electrical Engineering. The University of Queensland, Brisbane, Australia
30. Lu R, Sadiq S (2006) Managing process variants as an information resource. In: Proc. BPM06, pp 426–431
31. Lu R, Sadiq S (2007) On the discovery of preferred work practice through business process variants. In: Proc. ER'07
32. Luo Z, Sheth A, Kochut K, Miller J (2000) Exception handling in workflow systems. Applied Intelligence 13(2):125–147
33. Malone T, Crowston K, Herman G (2003) Organizing Business Knowledge - The MIT Process Handbook. The MIT Press
34. Mendling J (2007) Detection and prediction of errors in epc business process models. PhD thesis, Vienna Univ. of Economics and Business Administration
35. Minor M, Schmalen D, Koldehoff A, Bergmann R (2007) Structural adaptation of workflows supported by a suspension mechanism and by case-based reasoning. In: Proc. WETICE'07, pp 370–375
36. Minor M, Tartakovski A, Bergmann R (2007) Representation and structure-based similarity assessment for agile workflows. In: Proc. ICCBR'07, pp 224–238
37. Minor M, Tartakovski A, Schmalen D, Bergmann R (2008) Agile workflow technology and case-based change reuse for long-term processes. International Journal of Intelligent Information Technologies 1(4):80–98
38. Müller D, Herbst J, Hammori M, Reichert M (2006) IT support for release management processes in the automotive industry. In: Proc. BPM'06, LNCS 4102, pp 368–377
39. Müller D, Reichert M, Herbst J (2007) Data-driven modeling and coordination of large process structures. In: Proc. CoopIS'07, LNCS 4803, pp 131–149
40. Müller D, Reichert M, Herbst J (2008) A new paradigm for the enactment and dynamic adaptation of data-driven process structures. In: Proc. CAiSE'08, LNCS 5074, pp 48–63
41. Müller R, Greiner U, Rahm E (2004) AgentWork: A workflow system supporting rule–based workflow adaptation. Data and Knowledge Engineering 51(2):223–256
42. Mutschler B, Reichert M, Bumiller J (2008) Unleashing the effectiveness of process-oriented information systems: Problem analysis, critical success factors and implications. IEEE Transactions on Systems, Man, and Cybernetics (Part C) 38(3):280–291
43. Mutschler B, Weber B, Reichert M (2008) Workflow management versus case handling - results from a controlled software experiment. In: Proc. SAC'08, pp 82–89

44. Netjes M, Mansar S, Reijers H, van der Aalst W (2007) An evolutionary approach for business process redesign - towards an intelligent system. In: Proc. ICEIS (3), pp 47–54
45. Pande P, Neuman R, Cavanagh R (2000) The Six Sigma Way: How GE, Motorola, and Other Top Companies are Honing Their Performance. Mc-Graw Hill
46. Pesic M, Schonenberg M, Sidorova N, van der Aalst W (2007) Constraint-based workflow models: Change made easy. In: Proc. CoopIS'07, LNCS 4803, pp 77–94
47. Porter M (1985) Competitive Advantage: Creating and Sustaining Superior Performance. Free Press
48. Reichert M (2000) Dynamische Ablaufänderungen in Workflow-Management-Systemen. PhD thesis, Universität Ulm
49. Reichert M, Dadam P (1998) ADEPT$_{flex}$ – supporting dynamic changes of workflows without losing control. Journal of Intelligent Information Systems 10(2):93–129
50. Reichert M, Dadam P, Bauer T (2003) Dealing with forward and backward jumps in workflow management systems. Software and Systems Modeling 2(1):37–58
51. Reichert M, Rinderle S, Dadam P (2003) On the common support of workflow type and instance changes under correctness constraints. In: Proc. CoopIS'03, LNCS 2888, pp 407–425
52. Reichert M, Rinderle S, Kreher U, Dadam P (2005) Adaptive process management with ADEPT2. In: Proceedings ICDE'05, pp 1113–1114
53. Reijers H, Limam S, van der Aalst W (2003) Product-based workflow design. MIS 20(1):229–262
54. Reijers H, Mans R, van der Toorn R (2009) Improved model management with aggregated business process models. Data and Knowledge Engineering (to appear)
55. Rinderle S (2004) Schema evolution in process management systems. PhD thesis, University of Ulm
56. Rinderle S, Reichert M (2006) Data-driven process control and exception handling in process management systems. In: Proc. CAiSE'06, LNCS 4001, pp 273–287
57. Rinderle S, Reichert M, Dadam P (2004) Correctness criteria for dynamic changes in workflow systems – a survey. Data and Knowledge Enginnering 50(1):9–34
58. Rinderle S, Reichert M, Dadam P (2004) Disjoint and overlapping process changes: Challenges, solutions, applications. In: Proc. CoopIS'04, LNCS 3290, pp 101–120
59. Rinderle S, Reichert M, Dadam P (2004) Flexible support of team processes by adaptive workflow systems. Distributed and Parallel Databases 16(1):91–116
60. Rinderle S, Reichert M, Dadam P (2004) On dealing with structural conflicts between process type and instance changes. In: Proc. BPM'04, LNCS 3080, pp 274–289
61. Rinderle S, Weber B, Reichert M, Wild W (2005) Integrating process learning and process evolution - a semantics based approach. In: Proc. BPM'05, LNCS 3649, pp 252–267
62. Rinderle S, Reichert M, Jurisch M, Kreher U (2006) On representing, purging, and utilizing change logs in process management systems. In: Proc. BPM'06, LNCS 4102, pp 241–256
63. Rinderle S, Wombacher A, Reichert M (2006) Evolution of process choreographies in DYCHOR. In: Proc. CoopIS'06, LNCS 4275, pp 273–290
64. Rinderle S, Jurisch M, Reichert M (2007) On deriving net change information from change logs the DELTALAYER-algorithm. In: Proc. BTW'07, LNI P-103, pp 364–381
65. Rinderle-Ma S, Reichert M (2007) A formal framework for adaptive access control models. In: Journal of Data Semantics, IX, LNCS 4601, pp 82–112
66. Rinderle-Ma S, Reichert M (2008) Managing the life cycle of access rules in CEOSIS. In: Proc. EDOC'08, Munich, pp 257–266
67. Rinderle-Ma S, Reichert M, Weber B (2008) On the formal semantics of change patterns in process-aware information systems. In: Proc. ER'08, LNCS 5231, pp 279–293
68. Rinderle-Ma S, Reichert M, Weber B (2008) Relaxed compliance notions in adaptive process management systems. In: Proc. ER'08, LNCS 5231, pp 232–247
69. Rosemann M, van der Aalst W (2007) A configurable reference modelling language. Information Systems 32(1):1–23
70. Rummler GA, Brache AP (Year) Improving Performance: How to Manage the White Space in the Organization Chart. Jossey-Bass Publishers
71. Russell N, ter Hofstede A, Edmond D, van der Aalst W (2004) Workflow data patterns. Tech. Rep. FIT-TR-2004-01, Queensland University of Technology
72. Russell N, ter Hofstede A, Edmond D, van der Aalst W (2004) Workflow resource patterns. Tech. Rep. WP 127, Eindhoven Univ. of Technology
73. Russell N, van der Aalst W, ter Hofstede A (2006) Exception handling patterns in process-aware information systems. In: Proc. CAiSE'06, pp 288–302
74. Sadiq S (2000) Handling dynamic schema changes in workflow processes. In: Proc. 11th Australian Database Conference
75. Sadiq S, Sadiq W, Orlowska M (2005) A framework for constraint specification and validation in flexible workflows. Information Systems 30(5):349 – 378
76. SAP (2008) Sap solution maps: www.sap.com/solutions
77. Scheer AW (1994) Business Process Engineering: Models for Industrial Enterprises. Springer
78. Schonenberg H, Mans R, Russell N, Mulyar N, van der Aalst W (2008) Process flexibility: A survey of contemporary approaches. In: CIAO! / EOMAS, pp 16–30
79. Schonenberg H, Weber B, van Dongen B, van der Aalst W (2008) Supporting flexible processes through log-based recommendations. In: Proc. BPM'08, LNCS 5240, pp 51–66
80. SEI (2008) Capability maturity model: www.sei.cmu.edu/cmmi
81. Subramaniam S, Kalogeraki V, Gunopulos D, Casati F, Castellanos M, Dayal U, Sayal M (2007) Improving process models by discovering decision points. Information Systems 32(7):1037–1055
82. Thom L, Reichert M, Iochpe C (2009) Activity patterns in process-aware information systems: Basic concepts and empirical evidence. International Journal of Business Process Integration and Management (IJBPIM)
83. van der Aalst W (1998) The application of petri nets to workflow management. The Journal of Circuits, Systems and Computers
84. van der Aalst W, Jablonski S (2000) Dealing with workflow change: Identification of issues an solutions. Int'l Journal of Comp Systems, Science and Engineering 15(5):267–276
85. van der Aalst W, ter Hofstede A (2005) YAWL: Yet another workflow language. Information Systems 30(4):245–275
86. van der Aalst W, ter Hofstede A, Kiepuszewski B, Barros A (2003) Workflow patterns. Distributed and Parallel Databases 14(1):5–51
87. van der Aalst W, Weske M, Grünbauer D (2005) Case handling: A new paradigm for business process support. Data and Knowledge Engineering 53(2):129–162
88. van der Aalst W, Reijers H, Weijters A, van Dongen B, de Medeiros AA, Song M, Verbeek H (2007) Business

process mining: An industrial application. Information Systems 32(1):713–732
89. van der Aalst W, Dumas M, Gottschalk F, ter Hofstede A, la Rosa M, Mendling J (2008) Correctness-preserving configuration of business process models. pp 46–61
90. Vanderfeesten I, Reijers H, van der Aalst W (2008) Product based workflow support: A recommendation service for dynamic workflow execution. Tech. Rep. BPM-08-03, BPMcenter.org
91. Weber B, Reichert M (2008) Refactoring process models in large process repositories. In: Proc. CAiSE'08, LNCS 5074, pp 124–139
92. Weber B, Wild W, Breu R (2004) CBRFlow: Enabling adaptive workflow management through conversational CBR. In: Proc. ECCBR'04, LNCS 3155, pp 434–448
93. Weber B, Reichert M, Wild W, Rinderle S (2005) Balancing flexibility and security in adaptive process management systems. In: CoopIS'05, LNCS 3760, pp 59–76
94. Weber B, Reichert M, Rinderle S, Wild W (2006) Towards a framework for the agile mining of business processes. In: BPM'05 Workshop Proceedings, 1st Int'l Workshop on Business Process Intelligence (BPI'05) in conjunction with (BPM '05), LNCS 3812, pp 191–202
95. Weber B, Reichert M, Wild W (2006) Case-base maintenance for CCBR-based process evolution. In: Proc. ECCBR'06, LNCS 4106, pp 106–120
96. Weber B, Rinderle S, Reichert M (2007) Change patterns and change support features in process-aware information systems. In: Proc. CAiSE'07, LNCS 4495, pp 574–588
97. Weber B, Reichert M, Rinderle-Ma S (2008) Change patterns and change support features – enhancing flexibility in process-aware information systems. Data and Knoweldge Engineering 66(3):438–466
98. Weber B, Reichert M, Wild W, Rinderle-Ma S (2009) Providing integrated life cycle support in process-aware information systems. Int'l Journal of Cooperative Information Systems 18(1)
99. Weske M (2000) Workflow management systems: Formal foundation, conceptual design, implementation aspects. University of Münster, Habil Thesis
100. Weske M (2001) Formal foundation and conceptual design of dynamic adaptations in a workflow management system. In: Proc. HICSS-34
101. Weske M (2007) Business Process Management: Concepts, Methods, Technology. Springer
102. Xerox (1997) InConcert Technical Product Overview

Barbara Weber obtained her Ph.D. in Economics at the Institute of Information Systems, University of Innsbruck (Austria). Since 2004, she is researcher at the Department of Computer Science at the University of Innsbruck where she is currently working on her habilitation. Barbara is a member of the Quality Engineering (QE) research group and head of the research cluster on business processes and workflows at QE. Her main research interests are agile and flexible processes and intelligent user support in flexible systems. This spans several technology areas including workflow management systems, case-based reasoning, process-oriented knowledge management, enterprise information systems, process mining, and agile software development.

Shazia Sadiq is currently working in the School of Information Technology and Electrical Engineering at The University of Queensland, Brisbane, Australia. She is part of the Data and Knowledge Engineering (DKE) research group and is involved in teaching and research in databases and information systems. Shazia holds a PhD from The University of Queensland in Information Systems and a Masters degree in Computer Science from the Asian Institute of Technology, Bangkok, Thailand. Her main research interests are innovative solutions for Business Information Systems that span several areas including business process management, governance, risk and compliance, data quality management, workflow systems, and service oriented computing.

Manfred Reichert has been full professor at the University of Ulm since January 2008. From 2005 to 2007 he worked as Associate Professor at the University of Twente (UT) where he was coordinator of the strategic research initiatives on E-health (2005 - 2006) and Service-oriented Computing (2007). At UT he was also member of the Management Board of the Centre for Telematics and Information Technology which is the largest ICT research institute in the Netherlands. He has worked on advanced issues related to process management technology and service-oriented computing for ten years. Together with Peter Dadam he pioneered the work on the ADEPT process management system, which currently provides the most advanced technology for realizing flexible process-aware information systems. Manfred was PC Co-chair of the BPM08 conference in Milan and will be General Co-chair of the BPM09 conference in Ulm.