CSRD (2009) 23: 81–97
DOI 10.1007/s00450-009-0068-6

SPECIAL ISSUE PAPER

The ADEPT project: a decade of research and development for robust and flexible process support

Challenges and Achievements

Peter Dadam · Manfred Reichert

Published online: 22 April 2009

Abstract This paper gives insights into the ADEPT project. Its target was to develop a next generation process management technology, which is by orders of magnitudes more powerful and flexible than contemporary process management systems. The ADEPT technology should provide advanced features and properties within one system, which seem to exclude each other, but which are required for the support of a broad spectrum of processes: ease-of-use for end users and system developers, high flexibility through the support of non-trivial ad-hoc deviations at the process instance level, quick implementation of process changes through process schema evolution, and correctness guarantees enabling robust execution of implemented processes. This paper describes the background and the real-world cases which motivated our research. It further explains the technological challenges we faced, describes the solutions we elaborated, and discusses the current status of the ADEPT project.

Keywords Workflow Management · Business Process Management · Process Flexibility · Process Change · Correctness by Construction · Robustness

CR subject classification H.4.1 · D.2.2 · D.2.11

P. Dadam (✉) · M. Reichert
Institute of Databases and Information Systems,
University of Ulm,
James-Franck-Ring, 027,
89069 Ulm, Germany
e-mail: peter.dadam@uni-ulm.de

M. Reichert
e-mail: manfred.reichert@uni-ulm.de

1 Background – how all began

In 1992 we started the OKIS[1] project – a pretty large and broadly defined 3-years research project funded by the State of Baden-Württemberg. In this project several clinics, medical service units, and the computing center of our university hospital were involved. The goal of OKIS was to develop a concept for a cross-organizational, clinical information system that is able to integrate autonomous, heterogeneous departmental systems as well as to offer services across system boundaries (e.g., scheduling, resource management, and medical data exchange). At the beginning of OKIS we looked at many aspects of hospital information systems to understand the different types of relevant information, the different kinds of information systems involved (e.g., radiology information system, picture archiving and communication systems, and lab systems), the privacy issues, the different types and roles of doctors and nurses, characteristic properties of planning and scheduling tasks, and so forth. We further investigated new developments (at that point in time) like electronic patient record, communication servers, medical guidelines, and computer-assisted medical diagnostics in order to understand which features a modern hospital information system should have [27].

The challenge of the service layer we wanted to develop was rather clear in an early stage of the project: Due to the heterogeneity of the underlying systems and the evolutionary nature of the clinical domain one has to decouple service provision from service implementation. However, the discovery of services and their usage must not be complicated. Therefore, long before web services came into the game, we

[1] OKIS is derived from the German name of the project and stands for "Open Clinical Database and Information System for the Integration of Autonomous Subsystems".

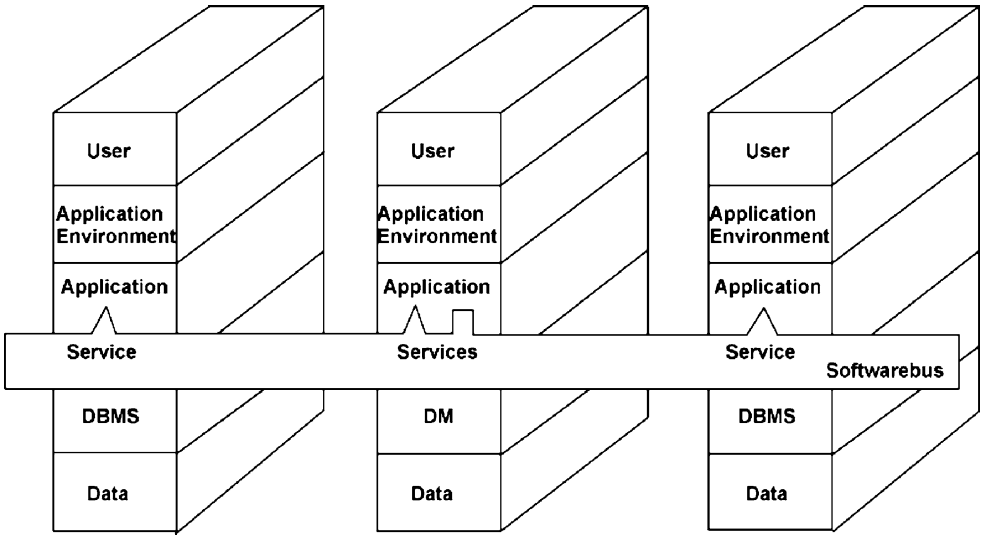

Fig. 1 OKIS software bus [28]

had elaborated the concept of a cross-organizational "service bus" (that we called "software bus" in [28]; see Fig. 1) into which new services can be easily plugged in such that application programs (providing the end-user interfaces) can use them.

While working on many different aspects of the overall problem we got the dim feeling that providing data integration together with some electronic services would be an improvement, but not be the big step forward our colleagues from the university hospitals were hoping for. Therefore, we had additional discussions with them as well as other clinical staff members, and also took a closer look at the clinical workday. During these activities it became more and more clear that the really big problem was not the inconvenient access to medical data. The much bigger and more challenging problem was the non-existing support of the clinical processes! These processes were only in the users' minds. Notes on paper or entries in a calendar system were the only help for physicians and nurses to not forget things. No active process support or assistance by an information system was provided to avoid problems like omission errors, unnecessarily pending tasks, or non-optimal task sequences (see [15] for a description of the problem). This meant that services provided by the *Software Bus* should be offered to users in a *process-oriented* way.

We got fascinated about this challenge and we were convinced that a software technology, which is able to cope with clinical processes, would be also adequate for many other domains and enable completely new perspectives for information management systems. Therefore, in 1995 we decided to start the ADEPT[2] project as a dedicated research activity in that subject area. At the beginning we were confronted with many problems and questions, not knowing where to start with and also not knowing which aspects were highly relevant for the final solution and which ones could be neglected. Nevertheless we made a decision which should guide and determine our whole research until today: *"We face the clinical reality – we do not define any problem away!"* This motto resulted in the insights, requirements, and technological challenges described in this paper.

Section 2 provides some insights into the clinical reality and identifies major requirements. Section 3 describes relevant challenges and the research areas of the ADEPT project. Section 4 discusses the overall technological challenges and the general "vision" of the ADEPT project. In Sect. 5 we present some of the achievements made. Section 6 describes the current development status and the transition from a research prototype to an industrial product. Finally, Sect. 7 concludes with a summary.

2 Facing the clinical reality

Our initial insights into relevant requirements were derived from the OKIS project. Following this, from 1996 to 1997 we performed a dedicated workflow project with Siemens-Nixdorf and our Women's Hospital. In this project we analyzed and documented the core processes of this hospital, investigated organizational aspects (e.g., actor responsibilities, substitution rules, or legal regulations), and evaluated what kind of exceptional cases had occurred in the past and how good they were "predictable". These insights were extremely helpful for us to extend and refine the requirements identified in the OKIS project, and to evaluate any suggested solution against these real-world scenarios [13]. The issues described in the sequel represent consolidated insights from both projects (and are valid until today).

[2] ADEPT stands for "Application Development based on Encapsulated Pre-modeled Process Templates"

Robustness. By nature, clinical information systems should be highly reliable. However, process-aware information systems (PAIS) are inherently more complex than traditional function- and data-centric information systems, simply because of the fact that the incorporated process support is another source of errors. In traditional information systems the processes are more or less only in the users' minds [39]. Of course, humans also make mistakes when performing processes, but these errors are typically not charged to the information system. However, once processes are directly supported by a PAIS, all process-related errors (e.g., deadlocks or program crashes due to missing input data) will now be charged to the PAIS and will directly affect its acceptance.

Flexibility and adaptivity of the clinical processes must not be restricted. In a clinical environment any PAIS will not be accepted by users if rigidity comes with it. Deviations from the standard procedure constitute the normal case, and physicians and nurses are accustomed to perform such deviations. The physician always has the ultimate professional authority and responsibility regarding decisions about diagnostic and therapeutic procedures. No computer program and thus no PAIS is allowed to overrule or to restrict the doctors' judgment. Being faced with this aspect it became clear that any kind of process technology that does restrict flexibility will fail in such a domain. However, the demand for flexibility is not only present in hospitals, it can be found in almost all domains (e.g., [4, 5, 32, 33, 35, 58]). No enterprise can take the risk to become inflexible, i.e., to be unable to quickly and flexibly react on changes in the market or in legal conditions, on detected inefficiencies in their processes, or on exceptional situations [35].

The support of clinical processes cannot be simply restricted to document-centered workflows, which would make the realization of a PAIS much easier, because any technological solution could then concentrate on control flow and would not have to deal with data flow issues.[3] Instead, we are faced with the full spectrum of process support ranging from simple form- or document-based, human-centric workflows to production workflows with manual activities, automatic activities, and need for application integration. In addition, runtime flexibility cannot be restricted to simple adjustments of a process schema (e.g., by replacing one activity by another), but more complex structural changes at the process instance level should be possible as well. For example, an ongoing treatment process might have to be changed to a large extent due to the physical reaction of the patient on his current medical drugs. However, such process flexibility must not lead to a high risk of PAIS failures at runtime or to a significant increase of the complexity when developing application functions. The great challenge for us was to find a solution which ensures a high degree of runtime flexibility on the one hand, and robustness as well as ease of use on the other hand.

Ease of use. Although listing this requirement may sound like the typical lip service, we considered ease of use as very important for a broad usage of process management technology in the clinical domain (and not only there). Clinical staff works under high time pressure, must often deal with exceptional situations, and is constantly confronted with an information overload [15, 29]. This situation especially applies to university hospitals which typically receive all complicated treatment cases that ordinary hospitals are not able to handle. In addition, university hospitals educate physicians and nurses; i.e., there are many staff members who are not very experienced. This, in turn, increases the pressure on clinical staff. Some staff members have stress because of their own missing routine and experience, while others suffer from stress because they have to supervise the less experienced colleagues in addition to their own duties. Therefore, any PAIS which increases this stress because of complicated handling will not be successful.

Ease of use must not only be achieved for end users, but should also hold for the developers of processes and corresponding application services. The problem is that ease of use for users does not come for free; i.e., somebody has "to pay the price". Supporting ad hoc changes at the process instance level, or changing a process schema at the process type level and propagating these changes to running instances, requires a profound understanding of basic PAIS concepts (e.g., correctness of process models) as well as deep knowledge about PAIS internals (e.g., the physical representation of process instances at the machine level). If such a detailed and system-near knowledge is required for process administrators or application programmers in order to avoid PAIS failures in the context of dynamic process instance changes, the battle will be lost before it will have begun.

3 Challenges

Taking all together, the ease of use aspect was probably the most influential one for our whole research. However, ease of use has different aspects and can be regarded from different perspectives: the end user, the process implementer, and the application developer. Our goal was to develop a technology which enables ease of use for all of them. Sections 3.1 to 3.3, therefore, focus on this aspect. Other important aspects, addressed in the ADEPT project as well, are discussed in Sect. 3.4 and 3.5.

[3] In this scenario there is no other data flow among process activities than the document which is passed from one activity to another during runtime.

3.1 Challenge: ease of use for process implementers

Ease of use for process implementers is influenced by several factors. An important one is how complicated it is to create a new process schema; i.e., which constructs and symbols are offered by the used process meta model, what is their semantics, how intuitive is their usage, and what kind of meta model related constraints have to be obeyed during process modeling? And it is also important that the process meta model is expressive enough. As another relevant factor process implementers should not need to know any implementation detail about the application functions the activities of a given process schema shall be associated with; i.e., for them, preferably, there should be no differences whether an application function is implemented as web service, Java library routine, or call interface to a legacy system. Instead, these application functions should all look like procedures or methods having input/output parameters. Finally, ease of use for process implementers is also significantly influenced by the implementation effort becoming necessary at their side in order to ensure that the composed process will be executable without runtime errors (e.g., concerning testing). From the very beginning it was clear for us that these factors must not exclude each other, but have to be considered in conjunction.

To speed up application development we pursued the idea of process composition in a *"plug & play"* style complemented by comprehensive correctness checks [8, 14] (cf. Fig. 2). Our target was to accomplish these checks in such a way that runtime errors during process execution can be excluded to a large extent. As prerequisite, data flows and other dependencies among application services, which are relevant for their execution order, must be somehow made known to the PAIS to be incorporated into the correctness checks. From our practical experiences it further became clear that intuitively usable modeling constructs and automated correctness checks alone will be not sufficient. A too liberal process meta model may result in too many undetected (semantical) modeling errors when checking correctness.

Another important issue concerns changes at the process type level. In the clinical domain it is very important that applied treatment procedures reflect the prevailing state of medical knowledge. Therefore, it must be possible to adapt them when medical knowledge changes. Such changes may also affect ongoing cases, i.e., it must be possible to modify a process schema and to migrate its instances to the new schema (we denote this as *process schema evolution* [10, 49]). As motivated such change feature should be easy to use (for the process specialist), comprehensive schema changes must be possible, and correctness checks on the system level should ensure robust execution of the adapted process instances.

3.2 Challenge: ease of use for application developers

As typical for other domains, in a hospital we are confronted with legacy systems offering special application functions to users. These legacy systems are implemented based on different platforms, have tasks which require user involvement and such which can be automated, and differ in their system interfaces, user interfaces, and interaction styles. The challenge was to provide all these heterogeneous application functions in a homogenized form to process implementers in order to make process composition in a plug & play fashion a reality. Our vision was that the process template fully encapsulates the process with all its application functions and services, such that the process just has to be "plugged" into the PAIS runtime environment to be executable. In addition, any manual activity coming with a graphical (e.g. form-based) user interface should smoothly integrate itself into the user's desktop; i.e., any kind of "window over window over window" effect had to be avoided. – This vision of "encapsulated process templates" also gave the project its name: ADEPT = "*A*pplication *D*evelopment based on *E*ncapsulated pre-modeled *P*rocess *T*emplates".

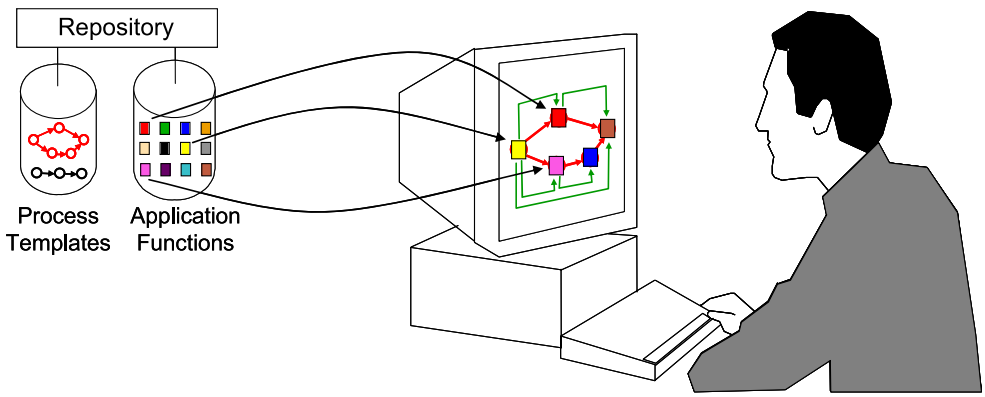

Fig. 2 Composition of processes using plug & play [11]

Ease of use for application developers meant to provide an easy to use implementation framework with intuitive application programming interfaces to perform all these tasks. Our maxim was to make the implementation of application components for a PAIS not more complicated than developing them for conventional application systems without process support. Particularly, all complexity coming along with the support of ad-hoc flexibility should not be put onto their shoulders.

3.3 Challenge: Ease of use for end users

Ease of use for end users includes adherence to typical human factors when designing a user interface; e.g., placement of information at the desktop, arrangement of entry fields, use of buttons, or selection of colors. We also experimented a little bit with user interface design, but this was not our primary focus. Instead, we studied how to make ad-hoc deviations at the process instance level as simple as possible, such that – in principle – a doctor or nurse can perform them autonomously (supposed that they have the permission to do that [54, 60]). From the above explanations it should become clear that every solution approach that requires from users a deeper understanding of system internals (e.g., "process states" and "data flows") would have to fight with big acceptance problems. And such approach would completely fail if it had been the user's responsibility to ensure that their ad-hoc changes do not lead to any subsequent runtime errors in the execution of the modified process instance. No doctor or nurse would accept to take this risk!

An end user interface for ad-hoc changes has to provide a sufficient level of abstraction. Users should only express what they want to be changed, but it should be the PAIS' task to figure out how to do that (in case the change is admissible). Figure 3a–h illustrate how the interaction between PAIS and end user may look like, presuming that the user is able to understand the meaning of a simplified (i.e. abstracted) process graph. Regarding this example assume that during the execution of a process instance (e.g., the treatment of a certain patient under risk) an additional lab test becomes necessary. Assume that this has not been foreseen at process implementation time (cf. Fig. 3a). As a consequence, this particular process instance will have to be individually adapted if the change request is approved by the system. After pressing the "exception button" (cf. Fig. 3b), the user can specify the type of the intended ad-hoc change (cf. Fig. 3c). If an insert operation shall be applied, for example, the system will display the application functions that can be selected in the given context (cf. Fig. 3d). These can be simple or complex application services, interactive or automatic application functions, or even complete processes. Following this, the user simply has to state after which activities(s) in the process the execution of the newly added activity shall be started and before which activities(s) it shall be finished (cf. Fig. 3e). If the activity to be inserted requires additional data for its input parameters, the PAIS will have to suggest the insertion of an appropriate auxiliary task. Finally, the system checks whether or not resulting process instance adaptations are valid (cf. Fig. 3f and g). All the validations needed to avoid runtime errors in the sequel as well as the necessary adaptations of the process structure, data flow, and instance state should be completely performed by the PAIS. In addition, the PAIS should allow for "intelligent" adaptations. For example, in order to enable a maximal degree of freedom in executing the newly added task, it should be insertable in parallel to the activities which are located between the ones marked as "after" and "before" in Fig. 3e (except that data flow dependencies require a more restricted execution).

This example illustrates a rather simple user interface. If more sophisticated, knowledge-based user interfaces are needed (e.g. [19, 34, 65]) this dialog can be simplified or even omitted. However, our process studies also revealed that ad-hoc changes are not always that simple. In certain cases it is not sufficient to only replace one activity by another or to just add a single activity directly before or after the currently activated one. And it is also not predictable in advance which parts of the process may be affected by a change. Therefore, we must also enable comprehensive structural changes that may rearrange large parts of the process or even completely replace them. Such complex changes are certainly beyond that what normal end users are able to do. Instead they require someone with appropriate knowledge in process modeling and process change. Such a person should have an interface which offers a comprehensive set of change operations. However, also in this case the PAIS should ensure robustness of the modified process instance.

In environments where exceptions and thus ad-hoc deviations occur rather frequently, it is often desirable to use a knowledge management system to support the user in detecting whether or not a similar exception already occurred in the past. Such a component should store which decisions were made with which success to solve the problem. By coupling it with the PAIS, it would become possible to conveniently "recycle" previous decisions and to automatically reapply changes at the process instance level [51, 61, 63, 65].

Not directly related to ease of use, but also important for end users are *response times* in connection with ad-hoc changes. Especially in the clinical domain, very likely, ad-hoc changes often will have to be performed under time pressure. Therefore, response times of the PAIS in the range of several seconds or even minutes (e.g., to decide on the correctness of an intended ad-hoc change) are not accept-

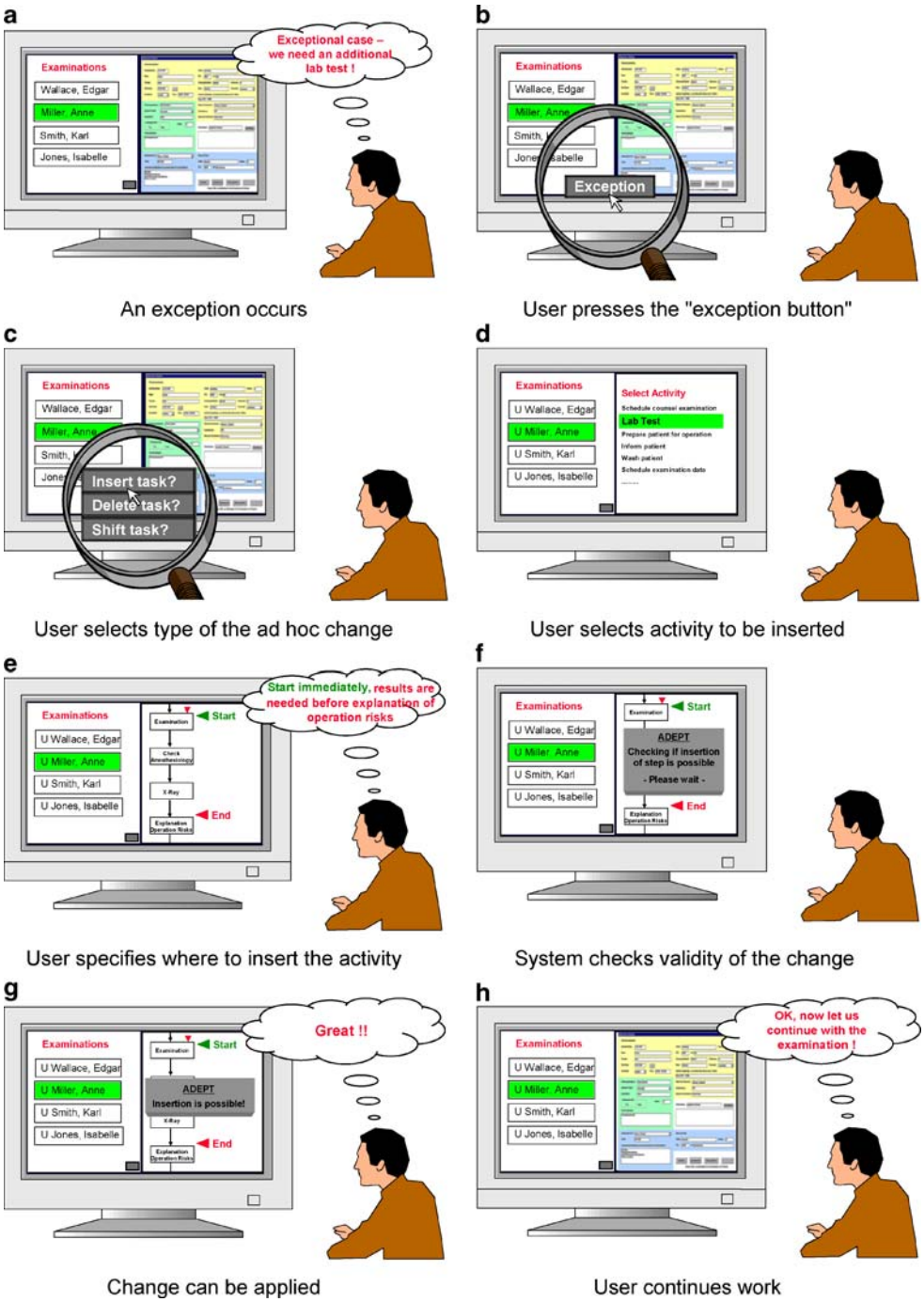

Fig. 3 Executing an ad hoc modification from the end-user's point of view

able. (Usually 3 seconds of response time are considered as upper limit for interactive tasks.) This means that the resulting solution must also use efficient algorithms for adapting process instances and for checking their correctness. Further, they must ensure short response times for scenarios in which many process instances are simultaneously executed by the PAIS. And the latter will be the normal case in large-scale environments where thousands of process instances are concurrently executed at the same time [7].

3.4 Complex ad-hoc changes and schema evolution

As discussed in Sect. 3.3 a PAIS should allow to handle all kinds of exceptional situations. In order to enable this without need for circumventing the PAIS, arbitrarily complex ad-hoc changes at the process instance level must be possible; e.g., authorized users should be allowed to move activities or whole process fragments to another position in the process graph.

Another important change aspect is to enable *process schema evolution* at the type level. Like other companies, hospitals are continuously adapting their organizational structures, are changing staff responsibilities, are outsourcing or insourcing tasks, and so forth in order to improve their (business) processes or to adequately react on changes in legislation or market demands. Many of these changes directly affect the processes supported by their (clinical) PAIS, i.e., these processes have to be adapted accordingly. While in some cases only simple attribute changes are required (e.g., to adapt the staff assignment rule of an activity), in others complex structural changes of the process schema become necessary. In case of short-running processes, usually, it is sufficient to finish the already started process instances according to the old process schema, while new process instances refer to the new schema. However, at the presence of long-running processes (months up to years) or in case of important and pressing changes this is not sufficient. Then it must be possible to perform *process schema evolution*, i.e., to *migrate* the process instances to the new schema version. Note that this also includes process instances which have been individually modified. – This is typical for today's environments where processes are executed manually; it will be hard to accept for companies if a PAIS does not support that. Both, complex ad-hoc changes and process schema evolution must be easy to accomplish for process experts. They should not require deep knowledge about system internals and not require to modify processes at a low level of abstraction.

3.5 Further requirements and challenges

To stay focused this paper concentrates on the technological challenges related to the described ease-of-use aspects showing how they influenced our solutions and how we dealt with the constraints we had to obey. In order to give a somewhat more complete picture of the problem domain, however, we want at least briefly mention some other challenges we were also confronted with.

Hospitals and especially university hospitals are large enterprises comprising many specialized clinics, having a large number of employees (often several thousand), and being confronted with thousands of "cases" (i.e. patients) which must be handled simultaneously. Therefore, any PAIS which does not scale up and which does not work in a cross-organizational setting will fail in such an environment. As the clinics of a university hospital may be geographically dispersed, it is rather unrealistic to assume that a centralized PAIS will always be the appropriate solution. Instead, concepts for the flexible, distributed execution of processes had to be found [6, 7, 37, 40].

Hospitals are often confronted with patients having multiple injuries, e.g., as a result of a traffic accident. Assume that such a patient has a broken leg and some injury to his head which have to be treated. It is not very likely that a process schema exists to handle these two injuries in common. Instead, a process schema for handling injured extremities and another for dealing with head injuries may exist. Obviously, as they affect the same patient in this case, they cannot be executed completely independent from each other. Problems of this kind require concepts for inter-process coordination [22, 23].

Deadlines and temporal constraints play an important role in the clinical domain as well. Typically an appointment (e.g., for performing a surgery) is made for a certain day and time which requires some preparatory activities. These should be scheduled at the appropriate points in time and warnings should be given by the PAIS if processing of the remaining activities at normal speed would jeopardize the deadline. Or in preparation of an examination the administration of some drug might become necessary. This drug should be administered not too early and not too late to achieve the desired effects. Future PAIS should therefore incorporate appropriate support for temporal constraint management [15, 20].

Finally, there are other challenges we have dealt with, but which we cannot discuss in detail. They include issues related to the learning from changes [21, 30], the efficient representation of changes in adaptive PAIS [52, 53], the visualization of business processes [9], and the evolution of organizational models and corresponding access rules [31, 54, 55].

4 Technological challenges and our vision

The technological challenges elaborated in the previous sections can be summarized as follows: We wanted to develop a PAIS which is by order of magnitudes more powerful and flexible than contemporary PAIS are, and whose change features are easy to use for end users, process implementers, and application developers. This sounds like a contradiction in itself, because we all know: *"There ain't no such thing as a free lunch."* However, regarding the mentioned user groups for which ease of use shall be achieved, we can see that one party is missing: the implementers of the fundamental PAIS technology. And we had one shining example to follow which had enabled ease of use by hiding the complexity beneath the surface: relational database technology. On the one hand, it was the first database technology which made it possible to support at the system level automatic query optimization, data independence from physical storage structures (relations, indexes), and powerful transaction-based concurrency control. On the other hand, it offered a user interface (SQL) which was by orders of magnitudes easier to learn and to use than the database interfaces before. And this was possible because it was based on powerful theories (relational algebra, query optimization, concurrency control). – Our hope (and basic belief) was that we can

achieve a similar effect for PAIS if we are able to develop the adequate underlying theory.

Our ambition was to develop a technology for PAIS, which is broadly applicable, i.e., not only to simple administrative processes, but also to highly dynamic and complex ones (e.g., diagnostic and therapeutic processes [19, 24, 29, 34]). The challenge was to develop a technology which supports "correctness by construction" during process composition and which guarantees correctness in the context of ad-hoc changes at the process instance level. This challenge was probably the most influential one for the whole ADEPT project. It had significant impact on the development of the ADEPT process meta model as well as on our work on process flexibility and process adaptivity. It meant, in essence, the following:

1. We have to hide the inherent complexity of process-orientation (especially in conjunction with flexibility) as far as possible from system administrators and application programmers; i.e, we have to perform all complex things "beneath the surface" in the process management system.
2. We have to provide powerful, high-level interfaces to application programmers, based on which they can implement easy to use end user interfaces.

When developing the ADEPT process meta model we were in a dilemma. On the one hand our analyses had shown that clinical processes can be complex structured; e.g., comprising alternative/parallel branchings and loops. A process meta model should therefore provide appropriate concepts to represent these structures adequately, i.e., it should be expressive enough. On the other hand our goal was to enable comprehensive and efficient correctness checks during process modeling as well as in conjunction with ad-hoc instance changes. The available theoretical works on flexible processes at that time either required simple process models (e.g., without loops [1], or without considering data flow [1, 66]), or required expensive analyses to decide whether or not the desired ad-hoc change can be granted [17].

Expressiveness of a process meta model has two major aspects: One is the ability to model a large variety of control flow structures in terms of process patterns [59]. Another one is how easy the semantics of the meta model constructs or the resulting control flows can be understood. First experiences indicated that process modeling based on states and transitions (as used in Petri Nets for example) is not very easy to understand for end users (i.e., doctors and nurses in our case). Another issue was that this notation quickly leads to large process models due to many symbols. Opposed to that, Activity Nets [26] were much easier to understand, but this approach had other weaknesses, including the missing support of loops and the context-dependent execution semantics of nodes; e.g., depending on its context the syntactic symbol for an activity node may represent a normal (sequential) node, an XOR split/join, or an AND split/join. We also elaborated other formalisms (e.g., state and activity charts, rule based approaches), but considered them not being appropriate for our purpose. Altogether the procedure of defining the ADEPT process meta model was no easy task and lasted several months during which we evaluated many aspects and their impact on the meta model and vice versa. Most headaches were caused by two partially conflicting goals: expressiveness and formal verification. All ideas were evaluated against the clinical processes we had acquired in our hospital projects.

The resulting process meta model as illustrated in Fig. 4 (see [36, 41] for details) does not look very fancy at first glance. However, its "ingredients" were carefully selected and complement each other. Thus the process meta model

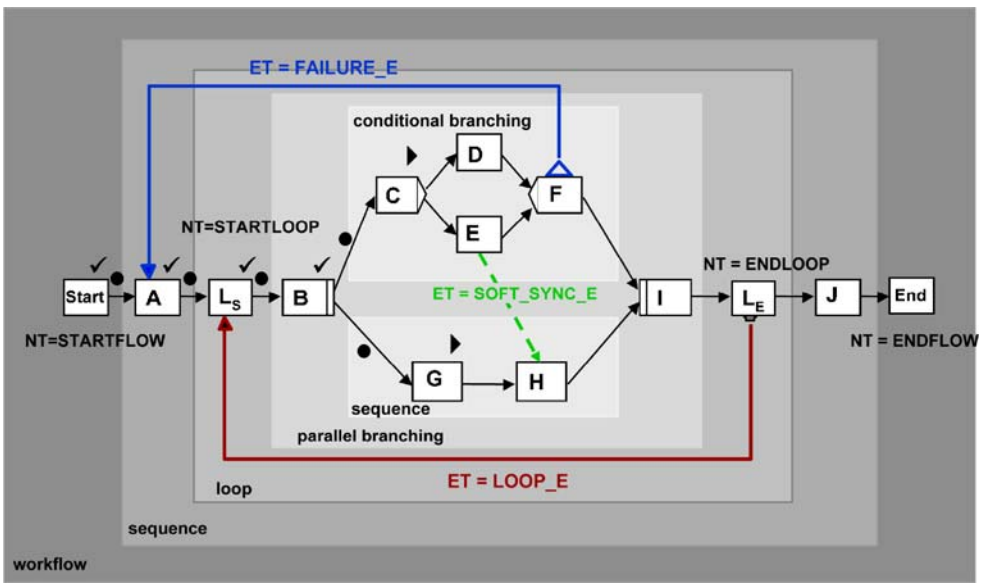

Fig. 4 ADEPT Process Meta Model

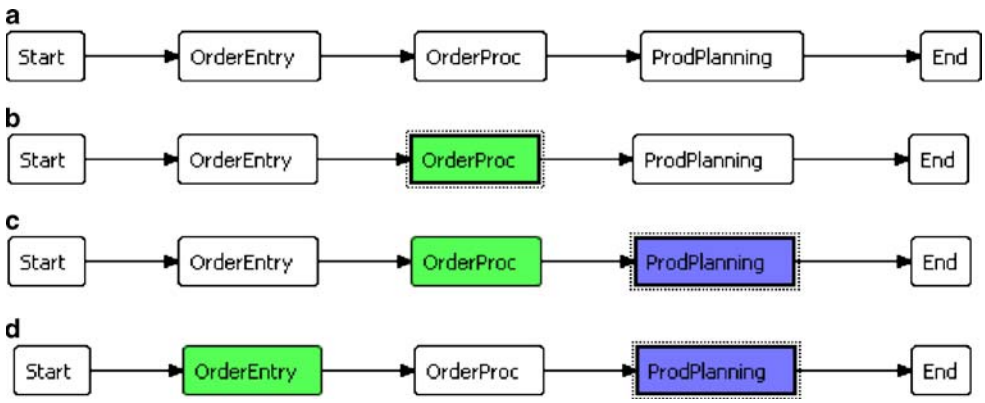

Fig. 5 Markings in the process graph

is very helpful with respect to formal verification, ad-hoc changes, and process schema evolution. Its strength is the underlying theory which supports both correctness by construction and efficient consistency checks [38, 64]. This theory precisely defines correctness criteria for the ADEPT meta model (e.g., absence of deadlocks, no isolated nodes, all data flows correct under all possible executions). It defines a comprehensive set of change operations with pre-/post-conditions which ensure that, if the desired change satisfies the preconditions, the resulting process schema will again be correct. The ADEPT change operations, for example, allow to serially insert an activity between two nodes, to insert it in parallel or between two node sets, to move activities, to delete activities, and so forth. All these operations obey that data flow correctness is not violated [36, 38].

Another important property of the ADEPT process meta model is that it incorporates not only the information on the current state at the instance level, but also information on how this state was reached. This allows to quickly decide whether a desired ad-hoc change can be granted or whether it affects an already passed region of the process instance. The latter could (among other things) cause data flow problems and is therefore prohibited (with some exceptions concerning loops [49]).

The block structuring of the process meta model was motivated by three aspects: First, experiments have shown that they are easier to handle and to understand for users when compared to unstructured process models. Second, it allows to restrict the area in the graph which has to be analyzed in the context of ad-hoc changes. This, in turn, helps to speed up the required analyses [38]. Third, it significantly simplifies the resulting structural adaptations of the process schema [64].

5 Achievements

The achievements described in the following refer to the ADEPT2 technology, and are structured along the challenges identified in Sect. 3.

5.1 Achievement: ease of use for process implementers

For process modeling, ADEPT2 provides an intuitive graphical editor. It applies a *correctness by construction* principle by providing at any time only those operations to the process implementer which allow to transform a structurally correct process schema into another one.

Operations are enabled or disabled according to which region in the process graph is marked for applying an operation. Figures 5 and 6 illustrate this relationship.

In Fig. 5a no nodes are marked. As a result, all operations in Fig. 6a are disabled, except `Insert Data Element` (which is not visible here). In Fig. 5b only activity "OrderProc" is marked and, therefore, those operations are enabled (cf. Fig. 6b) whose effects comply with this selection (e.g., to insert a surrounding AND block, a surrounding XOR block, a surrounding loop block, or to delete the selected activity). In addition, `Insert Data Element` (which is always applicable) is selectable again. In Fig. 5c two adjacent nodes are marked. The green color[4] indicates "begin of marked area" and the blue color[5] indicates "end of marked area". Again, those operations are enabled whose effect is precisely defined by such kind of marking (cf. Fig. 6c): In addition to the operations of the previous scenario, also the "in between" variants of the insertions are now enabled as well as the operation `Insert Node`. Finally, regarding the marking from Fig. 5d, at first glance, it might be astonishing that only the operations depicted in Fig. 6d (plus `Insert Data Element`) are enabled. However, only for these operations the effect is precisely defined for the given markings.

Deficiencies not prohibited by this approach (e.g., concerning data flow) are checked on-the-fly and are reported continuously in the problem window of the *Process Template Editor* (cf. Fig. 7). Another goal was to make the assignment of application functions to process steps as simple

[4] Light gray in a grayscale printout.
[5] Dark gray in a grayscale printout.

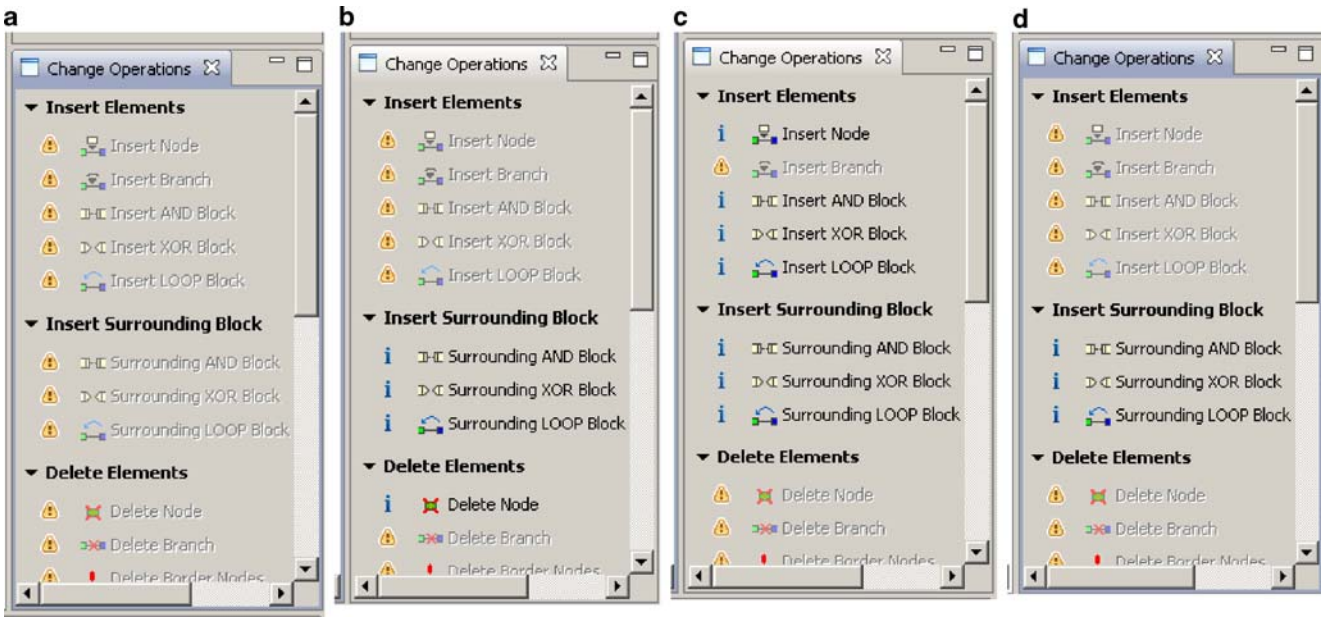

Fig. 6 Enabled change operations

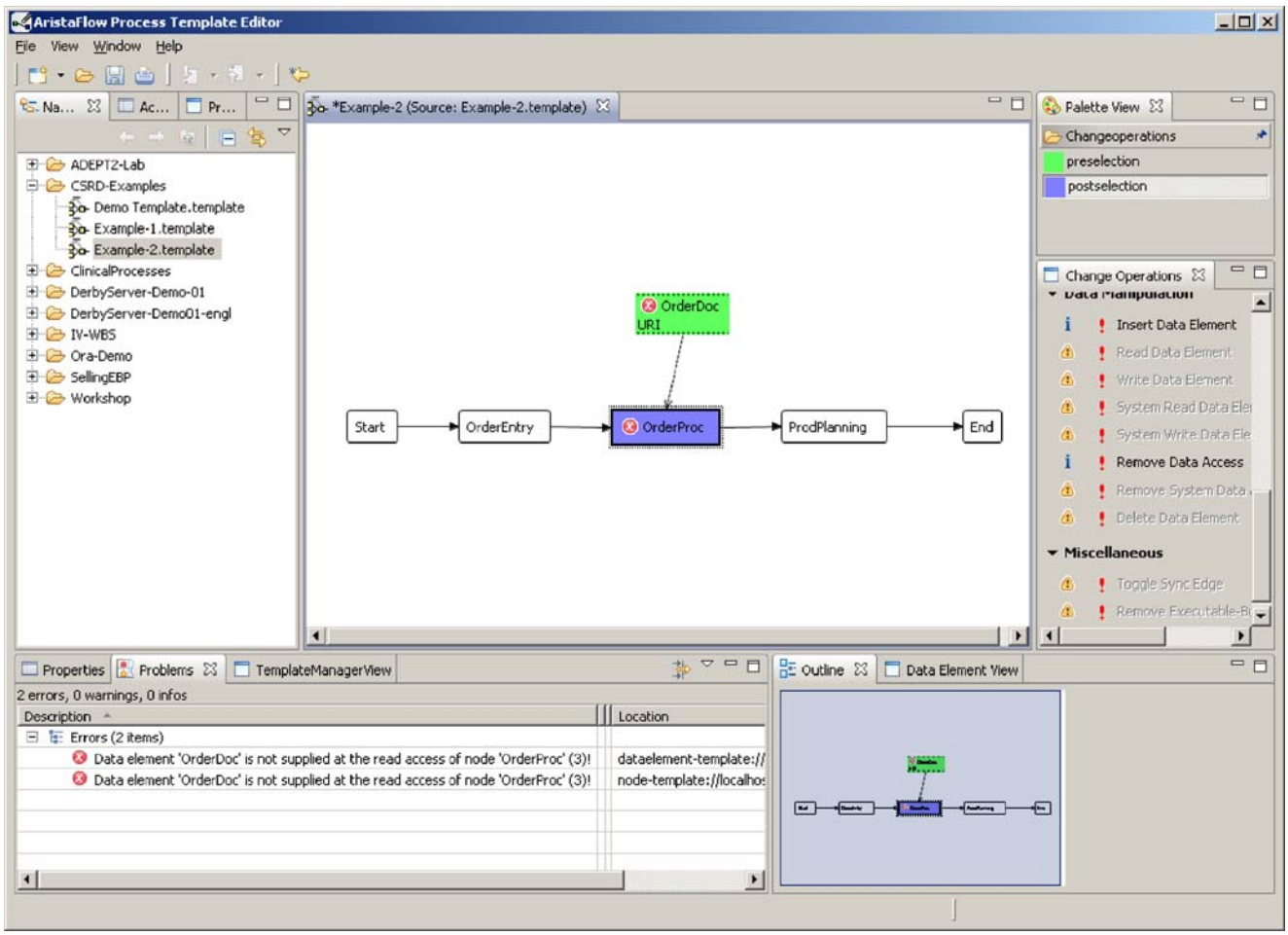

Fig. 7 Reporting of detected deficiencies (problem window on the *left* at the *bottom*)

as possible; i.e., a process implementer should not need to know details about the implementation of application functions. However, this should not be achieved by undermining the *correctness by construction* principle. Both goals have been achieved. All kinds of executables, that may be associated with process steps, are first registered in the *Activity Repository* as activity templates. An activity template provides all information to the *Process Template Editor* about mandatory or optional input and output parameters, as well as information about data dependencies to other activity templates. The process implementer just drags and drops an activity template from the *Activity Repository Browser* window of the *Process Template Editor* (cf. Fig. 8) onto the desired location in the process graph (cf. Fig. 2).

Depending on the intended purpose of usage, an activity template can be very specific or rather generic. When using a specific template everything can be fixed; e.g., the input and output parameters and all settings. In this case, the only remaining task for the process implementer is to check whether the proposed mapping of input/output parameters to process data elements (i.e., the process variables used within this process to communicate among activities) is correct. Using a specific database activity template, for example, allows to fix the input and output parameters, the details of the database used, the connection parameters, and the fully specified SQL statement. A more generic Activity Template, in turn, may leave open the SQL statement, the number and types of input and output parameters, or

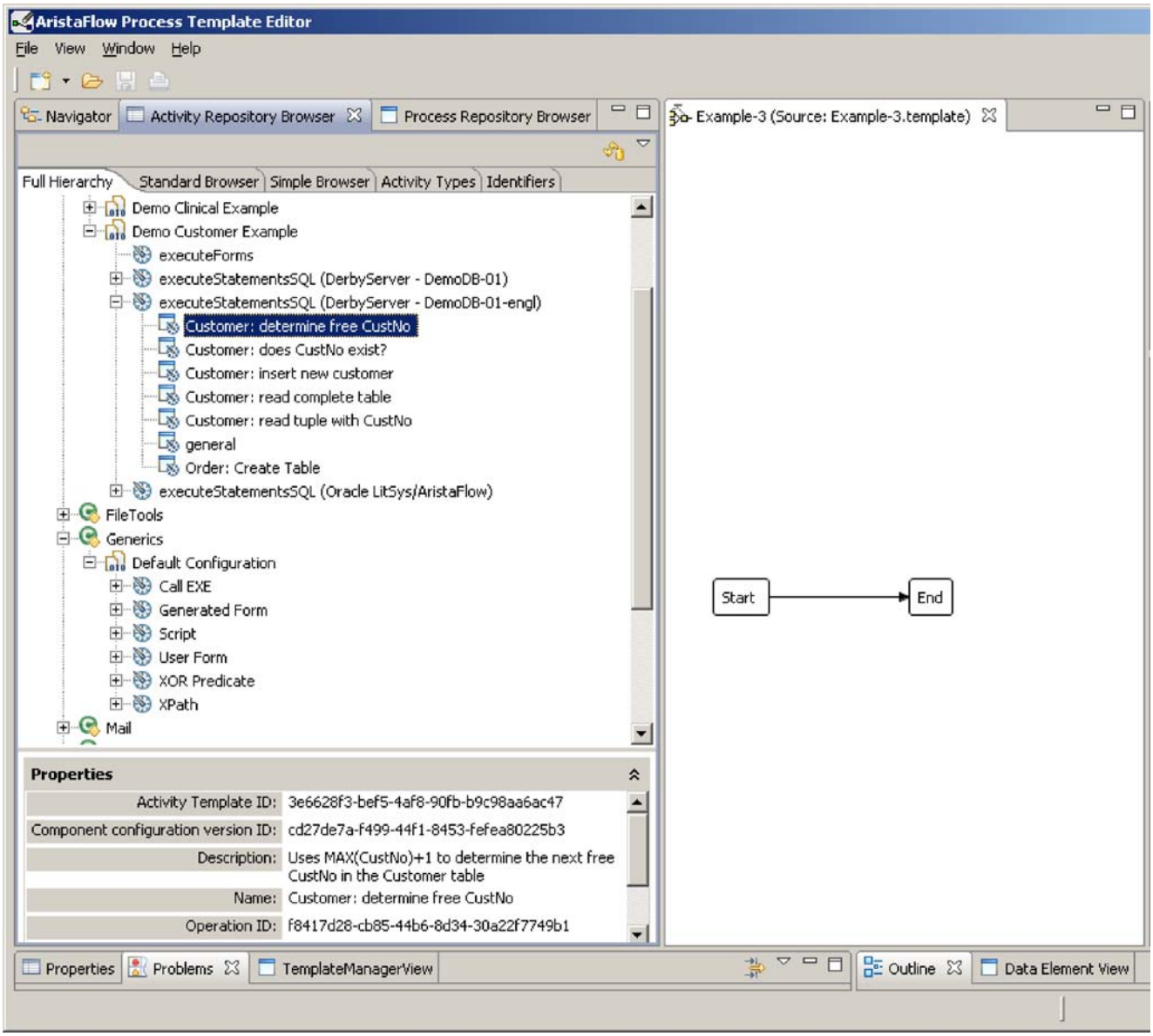

Fig. 8 Activity Repository Browser window in the Process Template Editor

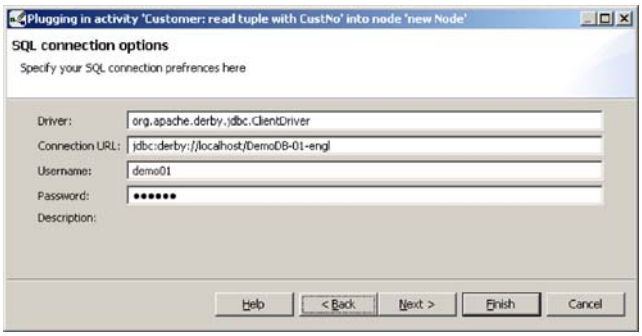

Fig. 9 Setting connection details in a DB activity template

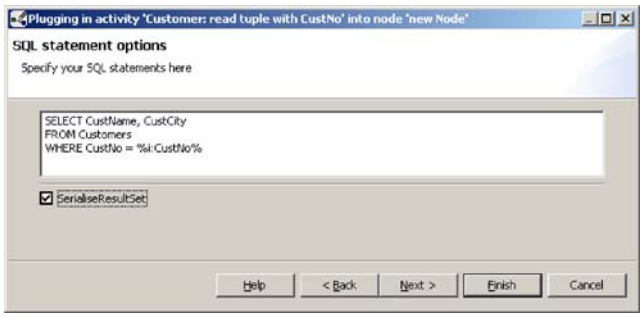

Fig. 10 DB activity template with defined SQL statement

the settings for the database connection (in parts or even completely).

5.2 Achievement: ease of use for application developers

As discussed in the previous section, all application functions are represented by activity templates; i.e., a developer who wants to provide a new application function or service must implement a corresponding activity template and add it to the *Activity Repository*. This makes it then available and accessible within the *ADEPT2 Process Template Editor* during process modeling (cf. Fig. 8). To simplify the implementation of such activity templates, ADEPT2 provides several levels of abstraction. At the lowest one ADEPT2 provides a so-called *Execution Environment* for each kind of basic operation supported by ADEPT2. For example, ADEPT2 offers execution environments for SQL statements, web services, EXE files, BeanShell scripts, basic file operations, and system-generated forms. However, the implementation of an execution environment requires some knowledge about ADEPT2 internals and, therefore, will typically not be the task of an ordinary application developer, but will be performed by system implementers.

An execution environment defines the set of methods needed to interact with the ADEPT2 runtime system as well as to implement the operations and facilities that shall be provided by the activity template. An activity template for database access, for example, may allow the user to specify connection details as illustrated in Fig. 9.

In general, the ADEPT2 runtime environment needs some information about the runtime behavior of the activities; e.g., whether or not they may be aborted, suspended, or undone. The implementer of an activity template has to implement interface methods that inform the ADEPT2 runtime environment which of these facilities are supported by the activity. For this case he must also provide the implementation of this functionality (see [45] for background information on ADEPT2 internals). The task of implementing a new activity template will be simple, if it can be based on a generic activity template. In this case, the implementation is essentially reduced to putting the appropriate entries into the set of forms representing the activity template. For example, if a database activity template shall be implemented which selects a tuple from a predefined relation in a predefined database based on a primary key value, one form will fix the required input parameters and the output parameters for the attribute values of the tuple, a second one the database connection (cf. Fig. 9), and a third one the SQL statement (cf. Fig. 10). Figure 8 lists examples of specialized activity templates which offer different kinds of operations on a customer table.

5.3 Achievement: ease of use for end users

To provide ease of use for end users is mainly the task of application developers. They decide how "manual" process activities interact with the end user. They also decide whether the standard ADEPT2 workflow client is used or whether a dedicated client shall be provided. An important prerequisite for realizing adequate user interfaces is to provide the appropriate methods to the application developer. In Fig. 3 we have shown how an interaction with the end user could look like when performing an ad-hoc change. To implement a workflow client with such capabilities, the application developer can make use of powerful system functions available at the ADEPT2 application programming interface (API):

- Querying the activity repository (using some filtering) for available activities.
- Marking the activity (or set of activities) after which the new activity shall become selectable.
- Retrieving from ADEPT2 the set of activities selectable as "end" activities for this insertion.
- Marking the activity (or set of activities) which shall serve as end activities.
- Performing (tentatively) the insertion based on this information.
- Checking the ADEPT2 report on detected errors (e.g. missing values for input parameter).
- Making the instance change persistent.

Using this API one can also implement domain-specific clients. In [19], for example, a knowledge-based approach was used to perform most of the process instance adaptations automatically without user interaction.

5.4 Achievement: Complex ad hoc changes and process schema evolution

Figures 11 and 12 illustrate how a non-trivial ad hoc change could look like. As example assume that a process instance wants to issue a request for a book quote using Amazon's web service facilities, but then fails in doing so. The user detects that his process instance is in trouble and calls the system administrator for help. The system administrator then invokes the *ADEPT2 Process Monitor* to take a look at this process instance (cf. Fig. 11). Looking into the execution log of the failed activity he detects that its execution failed because the connection to Amazon could not be established. Let us assume that he considers this as a temporary problem and offers the user to reset this activity so that it can be repeated once again. Being a friendly guy, he takes a short look at the process instance and its data flow dependencies, and sees that the result of this and the subsequent activity is only needed when executing the "Choose offer" activity. Therefore, he offers the user to move these two activities after activity "CheckSpecialOffers"; i.e., the user can continue to work on this process instance before the PAIS once again tries to connect to Amazon.

In order to accomplish this change he would switch to the *Instance Change Perspective* of the *Process Monitor* which provides the same set of change operations as the *Process Template Editor*. In fact, it *is* the *Process Template Editor*, but is aware that a process instance has been loaded and, therefore, all instance-related state information is taken additionally into account when enabling or disabling change operations and when performing correctness checks. The system administrator would now mark the two nodes "Get Amazon offer" and "Get Amazon price" as source area and the nodes "CheckSpecial Offer" and "Choose offer" as target area, and then perform the operation `Move nodes`. The resulting process graph is depicted in Fig. 12. Another option would be to move node "RetrieveSnailOffer" (where we are waiting for an E-Mail response) after "CheckSpecialOffer" as well. Then "CheckSpecialOffer" would become immediately selectable and thus executable.

Assume now that the web service problem lasts longer than expected and, therefore, the user may want to call Amazon by phone to get the price that way. In this case we would ask the system administrator to delete the two activities in trouble and to replace them with a form-based activity (which allows to enter the price manually and thus provides the value for the data element previously written by activity "Get Amazon price").

Regarding *process schema evolution* important goals were to provide the full spectrum of change operations for updating a process schema, to be able to migrate process instances (including those that were individually modified) to a new schema version (as far as possible), and to hide the inherent complexity of required checks and instance state adaptations as best as possible from the person in charge to perform this task. We invested a lot of energy into this subject in order to find a comprehensive solution to the problem [43, 46, 48, 50]. For the user (i.e., the process designer or process administrator), process schema evolution

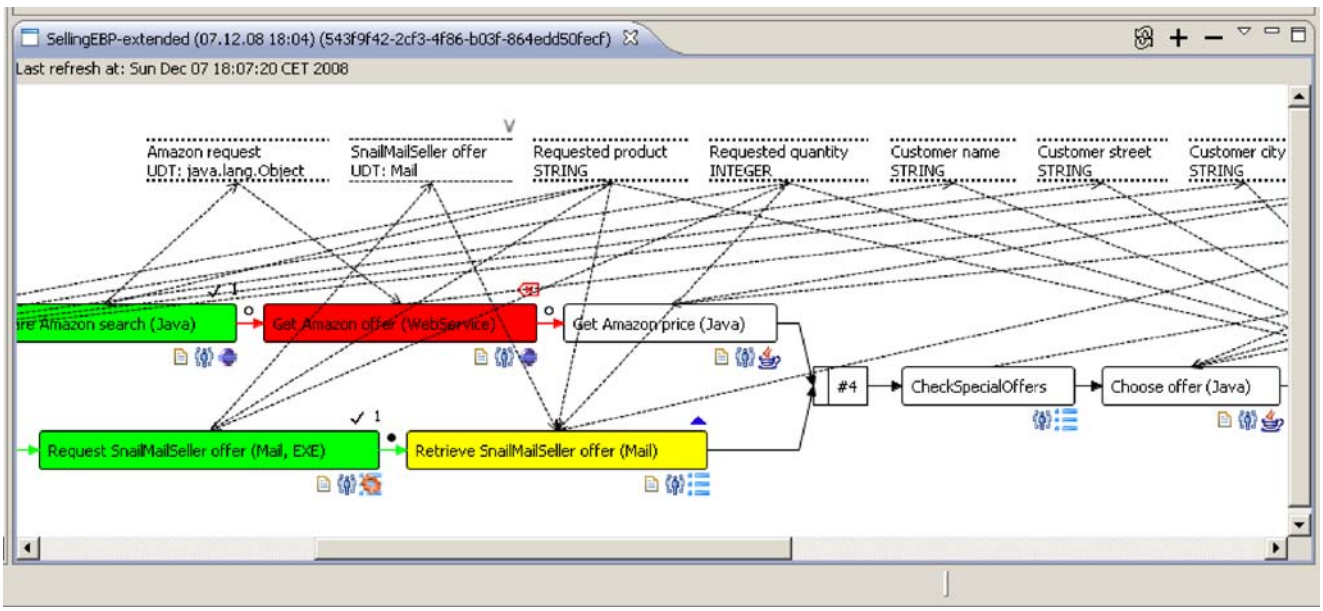

Fig. 11 Process monitor: monitoring perspective

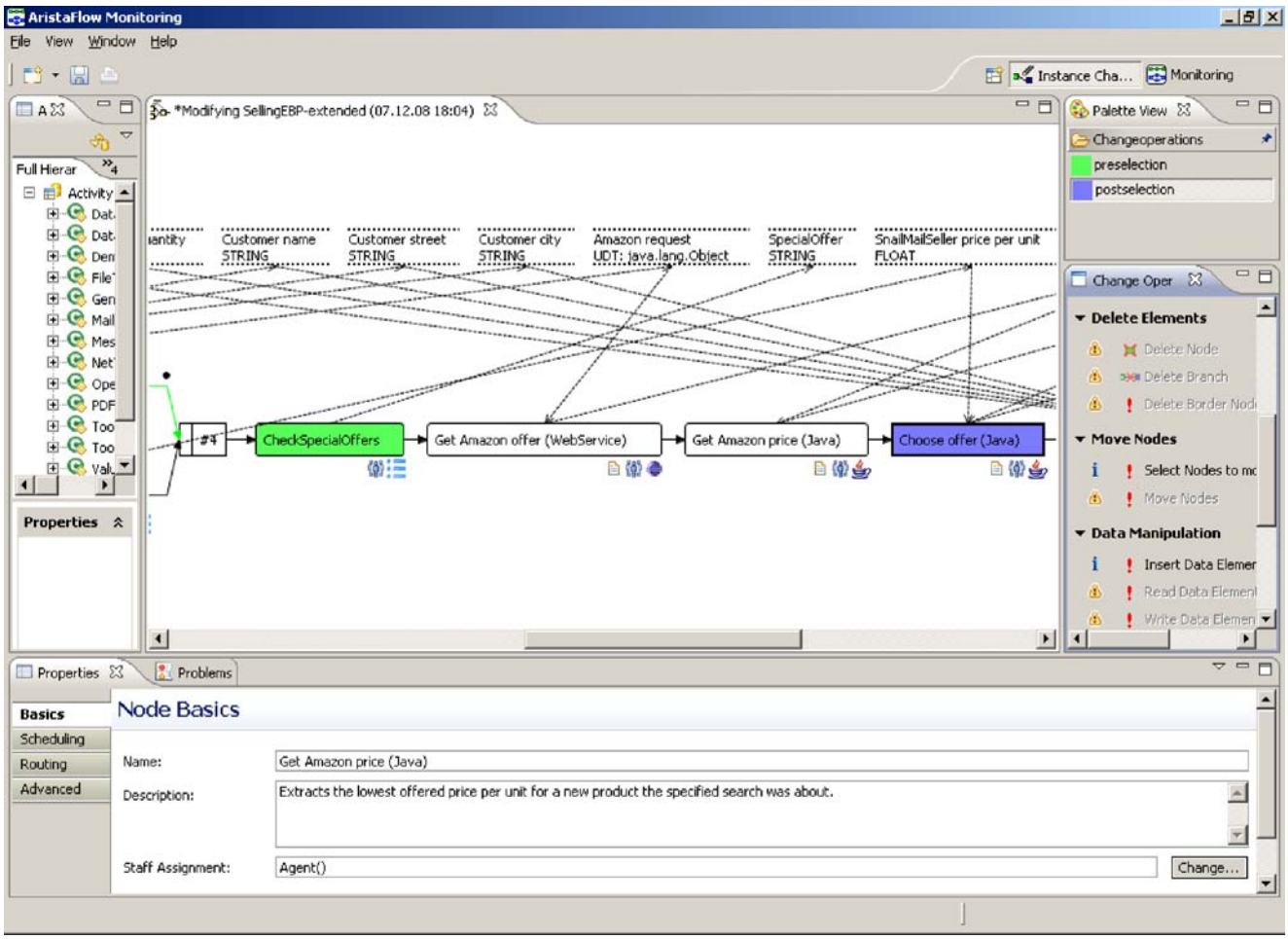

Fig. 12 Process monitor: instance change perspective

is nearly as simple as editing a process graph during ordinary process modeling. Like in the context of instance adaptations the *Process Template Editor* is invoked in a special mode such that it is aware that a new schema version is derived from an existing one. After a new schema version is derived, one can ask the ADEPT2 system to check which instances could be migrated to the new schema version and which not. These checks are based on well-defined compliance rules [49, 56]. Only if there is a rule which qualifies an instance as being "migratable", it is considered for migration, otherwise its execution continues on the old schema. For a more detailed description we refer to [16, 49].

6 Status of development: From ADEPT1 via ADEPT2 to AristaFlow® BPM Suite

In 1998 with ADEPT1 a first powerful prototype of the ADEPT technology became operational, which was then demonstrated at different events (e.g. [12, 25, 42]). It served as implementation base for process-oriented applications with manual as well as automated activities, and also provided some support for temporal constraints. Its most interesting feature, however, was certainly the robust support of ad-hoc deviations. ADEPT1 served as implementation platform for numerous projects (e.g. [4, 5, 18, 19, 60]) and was later extended to support distributed execution [6, 7, 40] as well (cf. Sect. 3.5).

In 2001 we intensified our research on process schema evolution which led to a first series of publications in 2003 [42, 43, 47]. As it would have been too much effort to modify the ADEPT1 code base in order to integrate these concepts, we developed a standalone proof-of-concept prototype [49]. In 2004 we received a research grant to perform a joint project with University of Mannheim and four industrial partners. The research project was named *AristaFlow* and was running until end of 2007. One project goal was to understand how the design and implementation of application functions could be supported by tools in such a way, that all necessary information to perform correctness checks during process modeling using plug & play (cf. Sects. 3.1 and 5.1) and ad-hoc deviations can be automatically de-

rived [2, 3]. The most important goal was to design and implement in parts the ADEPT2 process management system, which comprehensively supports all the functionalities developed in the ADEPT and AristaFlow project [44, 45].

The power of the ADEPT2 technology and the pre-versions demonstrating its capabilities attracted a number of companies. However, they could not base implementations of a real PAIS on an experimental system, especially if its maintenance and further development beyond 2007 was not assured. Therefore, at the beginning of 2008 we founded a spin-off (AristaFlow GmbH, Ulm) as joint venture with industrial partners to transfer ADEPT2 into an industrial-strength product version called *AristaFlow BPM Suite*, and to provide maintenance support for it. The screen shots used for illustrating ADEPT2 features in this paper have been taken from a pre-version of this product. The product version is now available for teaching and research purposes as well as for commercial applications.[6]

7 Summary

The research performed in the ADEPT project was motivated by problems we had identified in the clinical domain. This domain can be considered as "killer application" for PAIS, because one has to cope with conflicting goals: robustness, flexibility, and ease of use. To motivate the technological challenges we described our findings and insights from this domain in some detail. The most influential decision was to follow the motto: *"We do not define any problem away"*. We, therefore, never asked ourselves: *"Given a certain technology – what can we do with it?"*, but we asked instead: *"Given these real-world problems – which kind of technology is needed to adequately address them?"* At the beginning, ADEPT was a "high risk" research project because it was completely unclear whether or not this goal was achievable.

The resulting ADEPT technology has brought us further than we initially expected. Due to its "correctness by construction" principle, it allows to model, modify, and deploy processes very quickly. Its capabilities for ad-hoc deviation in conjunction with instantaneous checking of correctness does not only allow for the secure change of process instances, but also offers a complete new degree of freedom in modeling executable workflows. For example, one can start to execute only partially modeled processes and complement them during runtime. As example think of a project that will run three years. For many projects, it is probably not very attractive to model from the very beginning in great detail what shall be performed in the third year. One can go even further and, by starting with an empty process template, compose process instances on-the-fly (and have nevertheless all the full support of the underlying process management system). Pre-modeling all possible exceptions one can think of makes the process graph very complex and may allow execution paths which are undesired. With the ADEPT2 technology, one could separate exception handling from normal processing and use a knowledge-based system to modify process instances only on demand [51, 61, 62].

For both ADEPT2 and AristaFlow, much effort has been undertaken to make the API very powerful, but also easy to use. Experiences with first applications implemented on the new platform and utilizing the provided change capabilities make us confident that we have achieved this goal [57, 58]. We believe that ADEPT2 and AristaFlow show the capabilities, process technology will have to offer in future to be broadly applicable. It shows also that robustness, flexibility, and ease of use can be achieved in conjunction with each other.

Acknowledgement The described achievements would not have been possible without support from the partners in our various research projects, the staff members of our institute, and the numerous students who contributed to the ADEPT project. We mention just a few of them whose contribution had some impact on the further development and research directions of the project. We start with Klaus Kuhn, Birgit Schultheiß, and Stephan Frank who contributed a lot to our understanding of the clinical domain. As described in this paper, it were these insights which significantly influenced the whole project. Thomas Bauer addressed scalability issues and elaborated the fundamentals for the distributed version of ADEPT1. Clemens Hensinger did an extraordinary job the design and implementation of ADEPT1. The research of Stefanie Rinderle-Ma was another big step forward. Her contributions to process schema evolution extended the power of the ADEPT technology significantly. Last but not least we are indebted to Ulrich Kreher, Kevin Göser Martin Jurisch, and Markus Lauer for their great job in designing and implementing ADEPT2 and transferring this technology into an industrial-strength product.

References

1. Agostini A, De Michelis G (1998) Simple workflow models. In: Proc. of the Workshop on Workflow-Management, Lissabon, pp 146–163
2. Atkinson C, Stoll D, Acker H, Dadam P, Lauer M, Reichert M (2006) Separating per-client and pan-client views in service specification. In: Proc. IW-SOSE'06, pp 47–52
3. Atkinson C, Brenner D, Falcone G, Juhasz M (2008) Specifying high-assurance services. IEEE Comput 8:64–70
4. Bassil S, Benyoucef M, Keller R, Kropf P (2002) Addressing dynamism in e-negotiations by workflow management systems. In: Proc. DEXA'02
5. Bassil S, Keller R, Kropf P (2004) A workflow-oriented system architecture for the management of container transportation. In: Proc. BPM'04, pp 116–131
6. Bauer T, Dadam P (2000) Efficient distributed workflow management based on variable server assignments. In: Proc. CAiSE'00, Stockholm, pp 94–109

[6] The AristaFlow BPM suite is provided free of charge to universities for research and educational purposes. Please visit www.AristaFlow-Forum.de for more information on that. For commercial usage please visit www.AristaFlow.com.

7. Bauer T, Reichert M, Dadam P (2003) Intra-subnet load balancing in distributed workflow management systems. Int J Coop Inform Syst 12(3):295–323
8. Blaser R (1996) Configuration of distributed applications based on prefabricated program building blocks. Master's thesis, University of Ulm, DBIS Instiute (in German)
9. Bobrik R, Reichert M, Bauer T (2007) View-based process visualization. In: Proc. BPM'07, LNCS 4714, pp 88–95
10. Casati F, Ceri S, Pernici B, Pozzi G (1998) Workflow evolution. Data Knowl Eng 24(3):211–238
11. Dadam P (1997) Business information systems: Trends and technological challenges. In: Proc. BIS'97, pp 509–524
12. Dadam P, Reichert M (1998) The ADEPT WfMS project at the University of Ulm. In: Proc. 1st European Workshop on Workflow and Process Management (WPM'98), Zurich
13. Dadam P, Reichert M (2000) Towards a new dimension in clinical information processing. In: Proc. Medical Informatics Europe Conference (MIE'00), pp 295–301
14. Dadam P, Kuhn K, Reichert M, Beuter T, Nathe M (1995) ADEPT: An integrated ap-proach for the development of flexible, reliable, cooperating asssistant systems for the clinical domain. In: Proc. Annual Meeting of the German Informatics Society (Informatik'95), pp 677–686
15. Dadam P, Reichert M, Kuhn K (2000) Clinical workflows – the killer application for process-oriented information systems? In: Proc. BIS'00, Poznan, pp 36–59
16. Dadam P, Reichert M, Rinderle S, Jurisch M, Acker H, Göser K, Kreher U, Lauer M (2008) Towards truly flexible and adaptive process-aware information systems. In: Proc. UNISCON'08, LNBIP 5, pp 72–83
17. Ellis C, Maltzahn C (1997) The Chautauqua workflow system. In: Proc. Int'l Conf. on System Science, Maui, Hawaii
18. Golani M, Gal A (2006) Optimizing exception handling in workflows using process restructuring. In: Proc. BPM'06, LNCS 4102, pp 407–413
19. Greiner U, Müller R, Rahm E, Ramsch J, Heller B, Löffler M (2000) AdaptFlow: Protocol-based medical treatment using adaptive workflows. Methods of Information in Medicine, pp 80–88
20. Grimm M (1997) Adept-time: Temporal aspects in flexible workflow management systems. Master's thesis, University of Ulm, DBIS Instiute (in German)
21. 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 J Bus Proc Integr Manage, Spec Issue Bus Proc Flex 3(1):61–78
22. Heinlein C (2001) Workflow and process synchronization with interaction expressions and graphs. In: Proc. ICDE'01, pp 243–252
23. Heinlein C (2002) Synchronization of concurrent workflows using interaction expressions and coordination protocols. In: Proc. Confederated Int'l Conf. CoopIS'02, DOA'02, and ODBASE'02, LNCS 2519, pp 54–71
24. Heinlein C, Kuhn K, Dadam P (1994) Representation of medical guidelines using an clas-sification-based system. In: Proc. CIKM '94, pp 415–422
25. Hensinger C, Reichert M, Bauer T, Strzeletz T, Dadam P (2000) Adeptworkflow – advanced workflow technology for the efficient support of adaptive, enterprise-wide processes. In: Proc. EDBT'00 Software Demonstration Track, Constance, Germany, pp 29–30
26. IBM (1996) Workflow and Image Library: FlowMark and VisualInfo with Windows. SG24-4712-00
27. Kuhn K, Reichert M, Nathe M, Beuter T, Dadam P (1994a) An infrastructure for cooperation and communication in an advanced clinical information system. In: Proc. 18th Ann. Sym. on Computer Applications in Medical Care 1994, (SCAMC '94), pp 519–523
28. Kuhn K, Reichert M, Nathe M, Beuter T, Heinlein C, Dadam P (1994b) A conceptual approach to an open hospital information system. In: Proc. 12th Int'l Congress on Medical Informatics (MIE'94), pp 374–378
29. Lenz R, Reichert M (2007) IT support for healthcare processes – premises, challenges, perspectives. Data Knowl Eng 61(1):39–58
30. Li C, Reichert M, Wombacher A (2008) Discovering reference process models by mining process variants. In: Proc. ICWS'08, Beijing, pp 45–53
31. Ly T, Rinderle S, Dadam P, Reichert M (2005) Mining staff assignment rules from event-based data. In: Proc. BPM'05 workshops, LNCS 3812, pp 177–190
32. 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
33. 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
34. Müller R, Greiner U, Rahm E (2004) AGENTWORK: A workflow system supporting rule–based workflow adaptation. Data Knowl Eng 51(2):223–256
35. Mutschler B, Reichert M, Bumiller J (2008) Unleashing the effectiveness of process-oriented information systems: Problem analysis, critical success factors and implications. IEEE Trans Syst Man Cyb (Part C) 38(3):280–291
36. Reichert M (2000) Dynamische Ablaufänderungen in Workflow-Management-Systemen. PhD thesis, Universität Ulm
37. Reichert M, Bauer T (2007) Supporting ad-hoc changes in distributed workflow management systems. In: Proc. CoopIS'07, LNCS 4803, pp 150–168
38. Reichert M, Dadam P (1998) ADEPT$_{flex}$ – supporting dynamic changes of workflows without losing control. J Intell Inform Sys 10(2):93–129
39. Reichert M, Dadam P (2000) Geschäfts-prozessmodellierung und Workflow-Management: Konzepte, Systeme und deren Anwendung. Ind Manage 16(3):23–27 (in German)
40. Reichert M, Bauer T, Dadam P (1999) Enterprise-wide and cross-enterprise workflow management: Challenges and research issues for adaptive workflows. In: Proc. Workshop Informatik '99, CEUR 24, pp 56–64
41. Reichert M, Dadam P, Bauer T (2003a) Dealing with forward and backward jumps in workflow management systems. Softw Syst Model 2(1):37–58
42. Reichert M, Rinderle S, Dadam P (2003b) ADEPT workflow management system: Flexible support for enterprise-wide business processes. In: Proc. BPM'03, LNCS 2678, pp 370–379
43. Reichert M, Rinderle S, Dadam P (2003c) On the common support of workflow type and instance changes under correctness constraints. In: Proc. CoopIS'03, LNCS 2888, pp 407–425
44. Reichert M, Rinderle S, Kreher U, Dadam P (2005) Adaptive process management with ADEPT2. In: Proceedings ICDE'05, pp 1113–1114
45. Reichert M, Dadam P, Jurisch M, Kreher U, Göser K, Lauer M (2008) Architectural design of flexible process management technology. In: Proc. PRIMIUM Subconference at MKWI'08, CEUR 328, pp 415–422
46. Rinderle S (2004) Schema evolution in process management systems. PhD thesis, University of Ulm
47. Rinderle S, Reichert M, Dadam P (2003) Evaluation of correctness criteria for dynamic workflow changes. In: Proc. BPM'03, LNCS 2678, pp 41–57
48. Rinderle S, Reichert M, Dadam P (2004a) Disjoint and overlapping process changes: Challenges, solutions, applications. In: Proc. CoopIS'04, LNCS 3290, pp 101–120
49. Rinderle S, Reichert M, Dadam P (2004b) Flexible support of team processes by adaptive workflow systems. Distrib Parall Database 16(1):91–116

50. Rinderle S, Reichert M, Dadam P (2004c) On dealing with structural conflicts between process type and instance changes. In: Proc. BPM'04, LNCS 3080, pp 274–289
51. 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
52. 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
53. 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
54. Rinderle-Ma S, Reichert M (2007) A formal framework for adaptive access control models. J Data Semant IX:82–112, LNCS 4601
55. Rinderle-Ma S, Reichert M (2008) Managing the life cycle of access rules in CEOSIS. In: Proc. EDOC'08, Munich, pp 257–266
56. 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
57. Rüppel U, Wagenknecht A (2007) Improving emergency management by formal dynamic process-modelling. In: Proc. 24th Conf. on Information Technology in Construction (W78), pp 559–564
58. Rüppel U, Wagenknecht A (2008) Towards a process-driven emergency management system for municipalities. In: Proc. 12th Int'l Conf. on Computing in Civil and Building Engineering
59. van der Aalst W, ter Hofstede A, Kiepuszewski B, Barros A (2003) Workflow patterns. Distrib Parall Database 14(1):5–51
60. Weber B, Reichert M, Wild W, Rinderle S (2005a) Balancing flexibility and security in adaptive process management systems. In: CoopIS'05, LNCS 3760, pp 59–76
61. Weber B, Rinderle S, Wild W, Reichert M (2005b) CCBR–driven business process evolution. In: Proc. ICCBR'05, Chicago, pp 610–624
62. Weber B, Reichert M, Wild W (2006a) Case-base maintenance for CCBR-based process evolution. In: Proceedings ECCBR'06, LNCS 4106, pp 106–120
63. Weber B, Wild W, Lauer M, Reichert M (2006b) Improving exception handling by discovering change dependencies in adaptive process management systems. In: Business Process Management Workshops 2006, pp 93–104
64. Weber B, Reichert M, Rinderle-Ma S (2008) Change patterns and change support features – enhancing flexibility in process-aware information systems. Data Knowl Eng 66(3):438–466
65. Weber B, Reichert M, Wild W, Rinderle-Ma S (2009) Providing integrated life cycle support in process-aware information systems. Int J Coop Inform Syst 18(1)
66. Weske M (2001) Formal foundation and conceptual design of dynamic adaptations in a workflow management system. In: Proc. HICSS-34

Peter Dadam has been full professor at the University of Ulm and director of the Institute of Databases and Information Systems since 1990. Before he started his work at the University of Ulm he had been director of the research department for Advanced Information Management (AIM) at the IBM Heidelberg Science Center (HDSC). At HDSC he managed the AIM-P project on advanced database technology and applications. Current research areas include distributed, cooperative information systems, workflow management and database technology as well as their use in advanced application areas. Peter was PC Co-chair of the BPM07 conference in Brisbane, Australia. Together with Manfred Reichert he will be General Co-chair of the BPM09 conference in Ulm.

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.