CAiSE'06, Luxembourg, Luxembourg, June 2006, CAiSE Forum, Tool Demonstration (to appear)

ADEPT Next Generation Process Management Technology – Tool Demonstration

Manfred Reichert[1], Stefanie Rinderle[2], Ulrich Kreher[2],
Hilmar Acker[2], Markus Lauer[2], and Peter Dadam[2]

[1]Information Systems Group, University of Twente, The Netherlands
m.u.reichert@utwente.nl
[2] Dept. DBIS, University of Ulm, Germany
{stefanie.rinderle, ulrich.kreher, hilmar.acker, markus.lauer,
peter.dadam}@uni-ulm.de

1 Introduction

In the ADEPT project we have been working on the design and implementation of a next generation process management technology for several years [4, 8]. Based on a conceptual framework for dynamic process changes, on innovative process support functions, and on advanced implementation concepts, the developed system enables the realization of adaptive, process-aware information systems (PAIS). Basically, process changes can take place at the process type as well as the process instance level: Changes of single process instances [2, 4, 3] may have to be carried out in an ad-hoc manner (e.g., to deal with an exceptional situation) and must not affect system robustness and consistency. Process type changes, in turn, must be quickly accomplished in order to adapt the PAIS to business process changes [7, 5, 6]. This may also include the migration of (thousands of) instances to the new process schema (if desired). Important requirements are to perform respective migrations on-the-fly, to preserve correctness, and to avoid performance penalties.

2 Process Change Support

ADEPT offers powerful concepts for modeling, analyzing, and verifying process schemes. Particularly, it ensures schema correctness, like the absence of deadlock-causing cycles or erroneous data flows. This, in turn, constitutes an important prerequisite for dynamic process changes as well. In detail, ADEPT supports both ad-hoc changes of single process instances and the progagation of process type changes to running instances:

Ad-hoc changes of single instances: We support different kinds of ad-hoc deviations from the pre-modeled process template (e.g., to insert, delete, or shift activities). Such ad-hoc changes do not lead to an unstable system behavior, i.e., none of the guarantees achieved by formal checks at buildtime are violated due to the dynamic change. ADEPT offers a complete set of operations

for defining changes at a high semantic level and ensures correctness by introducing pre-/post-conditions for these operations. All complexity associated with the adaptation of instance states, the remapping of activity parameters, or the problem of missing data (e.g., due to activity deletions) is hidden from users.

Process type changes and change propagation: In order to deal with business process changes we enable quick and efficient schema adaptations at the process type level (schema evolution). In particular, it is possible to propagate type changes to running instances (of this type) as well. We provide a comprehensive correctness criterion for deciding on the compliance of process instances with a modified type schema. This criterion is independent of the used process meta model and is based on a relaxed notion of trace equivalence [7]. It considers control as well as data flow changes, and it works correctly in connection with loop backs. In order to enable efficient compliance checks, for each change operation we provide precise and easy to implement compliance conditions (cf. Fig. 1). Finally, efficient procedures exist for adapting the states of instance when migrating them to the new schema (cf. Instance I_1 in Fig. 1).

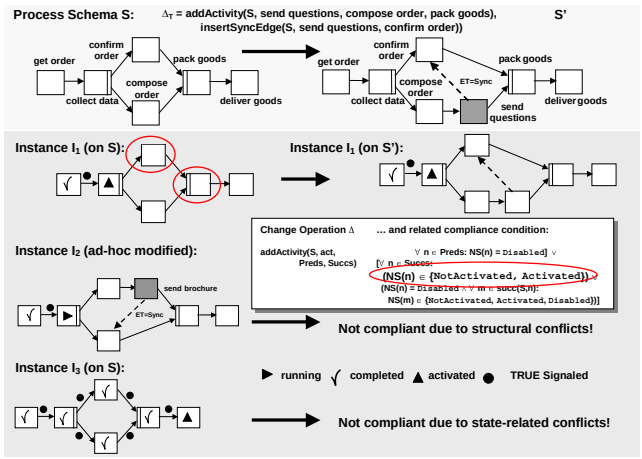

Fig. 1. *Process Schema Evolution and Change Propagation*

The correct interplay between concurrent type and instance changes is indispensable to provide real benefit for practical applications. Therefore, we have also dealt with the question how to propagate type changes to running instances that may be in different states and may have undergone preceding ad-hoc modifications. For such 'biased' instances, the current execution schema differs from the original one.We use a comprehensive correctness principle in this context, which excludes state-related, structural, and semantical conflicts. Fig. 1 shows an example: Instance I_2 has been individually modified such that type change Δ_T cannot be applied to it; otherwise the resulting instance schema would contain a deadlock-causing cycle.

ADEPT comprises a number of system components. The buildtime components support the correct modeling of process schemes and the process-oriented composition of application services in a plug & play like fashion. The runtime components, in turn, enable (distributed) process control and worklist management, support dynamic process changes at the described levels (also in case of distributed process control), and show how the different concepts also work in conjunction with each other. Furthermore, comprehensive interfaces for application programming are offered.

The implementation of ADEPT has raised many challenges, e.g., with respect to the representation of schema and instance data: Unchanged instances are stored in a redundant-free manner by referencing their original schema and by capturing instance-specific data (e.g., activity states). As an example, consider instances I_1, I_3, I_4, and I_6 from Fig. 2. For changed ('biased') instances, this approach is not applicable. One alternative would be to maintain a complete schema for each biased instance, another to materialize instance-specific schemes on-the-fly. We follow a hybrid approach: For each biased instance we maintain a minimal substitution block that captures all changes applied to it so far. This block is then used to overlay parts of the original schema when accessing the instance (I_2 + I_5 in Fig. 2).

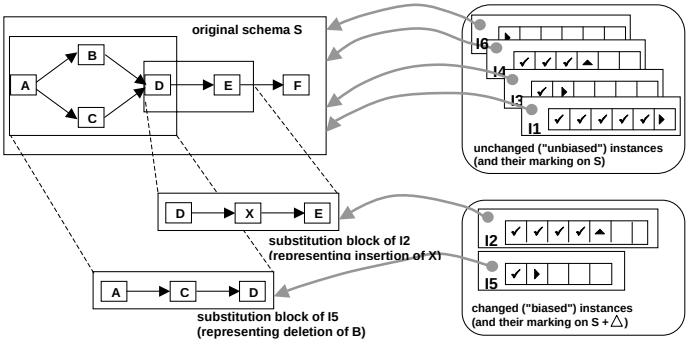

Fig. 2. *Managing Schema and Instance Data*

3 Demo description

In our prototype, effects of ad-hoc instance changes can be visualized by a special monitoring component. The same applies for process type changes (cf. Fig. 3). Users define a new process type by introducing a respective process template. Based on such a template new instances can be created and new schema versions be derived. Fig. 3 shows version V2 of an online ordering process. After committing the change, the system automatically checks compliance conditions and reports migration results to the user (cf. Fig. 3). This report summarizes which instances are compliant with the new schema version. For non-compliant instances the report indicates state-related (I_3 in Fig. 3) or structural conflicts (I_2 in Fig. 3). Fig. 3 also illustrates the interplay between type and instance

changes: I_2 has been individually modified, but cannot be migrated to the new schema version due to a structural conflict.

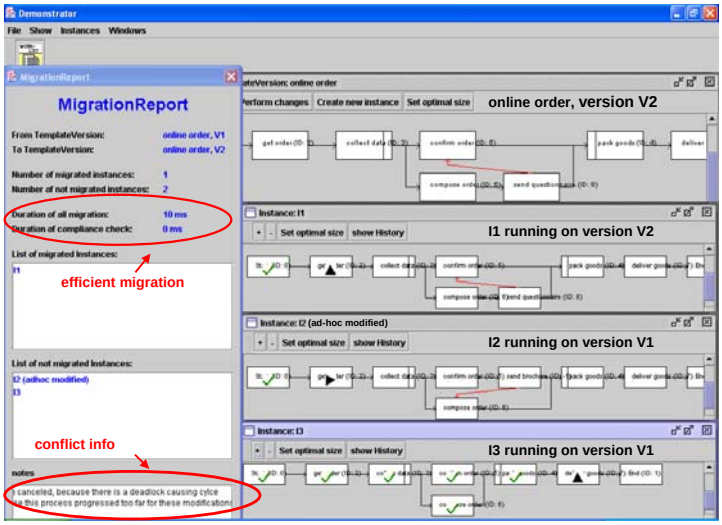

Fig. 3. *Screen of ADEPT Demonstrator*

In order to gain experience we have deployed this system to different research groups (e.g., [1]). They have used it as platform for realizing flexible PAIS in domains like healthcare, e-commerce, transportation, and the automotive industry. The experiences made have helped us to refine our conceptual framework and to develop new system components with advanced programming interfaces.

References

1. S. Bassil, R. Keller, P. Kropf: *A Workflow-oriented System Architecture for the Management of Container Transportation.* Proc. BPM'04, Potsdam, LNCS 3080, June 2004, pp. 116–131.
2. W.M.P.v.d. Aalst: Exterminating the Dynamic Change Bug: A Concrete Approach to Support Worfklow Change, Inf. Sys. Frontiers, 3:297-317, 2001
3. M. Reichert, P. Dadam, T. Bauer: *Dealing With Forward and Backward Jumps in Workflow Management Systems.* SoSyM, 2(1):37-58, 2003
4. M. Reichert, P. Dadam: *ADEPT$_{flex}$ – Supporting Dynamic Changes of Workflows Without Losing Control.* JIIS 10(2): 93–120 (1998).
5. C.A. Ellis, K. Keddara, G. Rozenberg: Dynamic Change within Workflow Systems. Proc. COOCS'95, 1995, Milpitas, CA, pp. 10-21
6. F. Casati, S. Ceri, B. Pernici, G. Pozzi: Workflow Evolution, DKE, 24:211-38, 1998
7. S. Rinderle, M. Reichert, P. Dadam: *Flexible Support Of Team Processes By Adaptive Workflow Systems.* Distributed and Parallel Databases, 16(1):91–116 (2004).
8. S. Rinderle, B. Weber, M. Reichert, W. Wild: Integrating Process Learning and Process Evolution. Proc. BPM'05, Nancy, France, 2005, 252–267