ADEPT Workflow Management System:[⋆]
Flexible Support for Enterprise-Wide Business Processes
– Tool Presentation –

Manfred Reichert, Stefanie Rinderle, and Peter Dadam

University of Ulm, Faculty of Computer Science,
Dept. Databases and Information Systems
{reichert, rinderle, dadam}@informatik.uni-ulm.de

Abstract. In this tool presentation we give an overview of the ADEPT workflow management system (WfMS), which is one of the few available research prototypes dealing with enterprise-wide, adaptive workflow (WF) management. ADEPT offers sophisticated modeling concepts and advanced features, like temporal constraint management, ad-hoc WF changes, WF schema evolution, synchronization of inter-workflow dependencies, and scalability. We sketch these features and describe how they have been realized within ADEPT. In addition, we show which tools and interfaces are offered to developers and users in this context. ADEPT follows a holistic approach, i.e., the described concepts have not been implemented in an isolated fashion only, but are treated in conjunction with each other by integrating them within one WfMS.

1 Introduction

Long regarded as technology for the automation of well-structured business processes, WF management is in the throes of transformation as more and more non-traditional applications require comprehensive process support. In many domains, like hospitals, engineering environments, or e-business, however, high requirements with respect to functionality, flexibility, and scalability exist [1,2,3]. In the ADEPT project, we have addressed these requirements from the very beginning. In the meantime, we have developed an adaptive WfMS prototype, which allows users to realize flexible, enterprise-wide WF applications.

In this paper, we give an overview of the ADEPT WfMS and its related concepts, tools, and user as well as programming interfaces. Section 2 summarizes basic features of the ADEPT WfMS, which have been described in more detail in previous publications of our group [2,4,5]. In Section 3 we show how these features have been realized within the ADEPT WfMS. Section 4 sketches selected projects to demonstrate the usefulness of the developed WfMS. We conclude with a short summary in Section 5.

[⋆] This work was partially performed within the research project "Change management in adaptive workflow systems", which has been founded by the German Research Community (DFG).

2 Features of the ADEPT WfMS

WF Modeling: ADEPT offers advanced concepts for the modeling, analysis, and verification of WF templates [5]. It enables the explicit definition of control and data flow, actor and resource assignments, temporal constraints, and pre-planned exceptions (e.g., forward and backward jumps [6]). This can be done in an integrated and consistent manner. Thereby ADEPT guarantees static and dynamic correctness properties (e.g., no missing input data when invoking activity programs, no undefined work assignments, no deadlocks), which is an important prerequisite for later model as well as instance changes. For control flow modeling, a simple, yet powerful formalism is offered. It is based on serial-parallel graphs with several important extensions necessary to adequately capture real-world processes. Nevertheless the resulting WF models are easy to understand for designers as well as for end users. In addition to this graph-based representation, a precise formal semantics, an equivalent operational semantics, and an efficient implementation exists.

Temporal Constraints: The handling of temporal constraints is an important feature of any WfMS. In ADEPT, designers can specify minimal and maximal durations for WF activities. At runtime, in addition, appointments may be associated with them. Furthermore, time dependencies between activities are definable (e.g., "`X` must be completed 2 days before `Y` starts"). ADEPT offers advanced concepts for specifying such constraints and for checking already at buildtime whether they are satisfiable or not [2]. Currently, we use *Temporal Constraint Networks* for representing time constraints and for checking consistency. At runtime, ADEPT schedules activities according to their starting times, supervises temporal constraints, and informs users when deadlines are going to be missed. Problems we have to deal with in this context include uncertainty, delays, and temporal inconsistencies (e.g., due to model changes).

Ad-hoc WF Changes: The support of ad-hoc changes is a must for WfMS in order to cover a broad spectrum of processes. At the instance level, ADEPT enables different kinds of ad-hoc deviations from the pre-modeled WF template (e.g., to omit activities, to change activity sequences, or to insert activities [5]). Such dynamic changes, however, must not lead to an unstable system behavior; i.e., none of the guarantees which have been achieved by formal checks at buildtime must be violated due to the dynamic change. ADEPT ensures this by introducing formal pre- and post-conditions for change operations. In particular, a consistent state must be preserved when a WF instance is going to be adapted. Additionally, ADEPT properly integrates changes with respect to authorization and documentation. Furthermore, all complexity associated with the adaptation of WF instance states, the re-mapping of input/output parameters of the components affected by a change, the problem of missing input data due to activity deletion, or the problem of deadlocks is hidden to a large degree from users.

WF Schema Evolution: In order to adequately deal with business process changes it is important that adaptations can be quickly performed at the WF type level as well. Besides versioning, ADEPT supports the propagation of WF type changes to in-progress WF instances. In doing so, change propagation is restricted to those WF instances for which the type change does not conflict with current instance state or previous ad-hoc changes. Basic to this is a comprehensive framework for change propagation which is based on well-defined compliance criteria for WF instances and on advanced rules for automatically and efficiently adapting instance markings.

Specification and Synchronization of Inter-WF-Dependencies: Many WfMS do not provide adequate means for (semantic) inter-workflow coordination as concurrently executed WF instances are considered completely independent. Though WF templates are modeled separately from each other in order to remain comprehensible and manageable, very often corresponding instances are semantically inter-related in the one way or another [4]. Pragmatical approaches like inter-workflow message passing or merging interdependent workflows within one template do not satisfactorily solve this problem. The latter, for example, would lead to a large number of templates, each of them very complex and hard to maintain. ADEPT uses *interaction expressions* and *interaction graphs* as a simple yet powerful mechanisms for the specification and implementation of inter-WF dependencies [4]. In addition to a graph-based semi-formal interpretation, a precise formal semantics, an equivalent operational semantics, an efficient implementation, and detailed complexity analyses exist, which allow us to actually apply this formalism to coordinate inter-WF dependencies. ADEPT uses different coordination and subscription protocols to actually employ interaction expressions for the efficient synchronization of concurrent workflows.

Scalability and Distributed WF Control: In large-scale, enterprise-wide application scenarios, performance is a critical issue. Due to the high amount of communication between server(s) and clients the communication network may become a bottleneck, especially if a large amount of "long-distance" communication occurs. To avoid bottlenecks, ADEPT allows to reduce the network load by partitioning WF graphs and by migrating the control of WF instances from one server to another during run-time [7,8]; i.e., a WF instance may no longer be controlled by only one WF server. When performing such a migration, a description of the instance state is transmitted to the target server. This includes information about activity states as well as WF relevant data. To avoid unnecessary communication between servers, ADEPT allows to control parallel branches of a WF instance independently from each other (at least as no synchronization due to other reasons, e.g., a dynamic WF change, becomes necessary).

When designing these features, the following issues have been of interest: How to maintain robustness and correctness, how does the feature affect application programming, and how is it made available to the end user? In addition, we have identified the interdependencies existing between them and we have shown how the different features work in conjunction with each other.

3 ADEPT Components, Architecture, and Interfaces

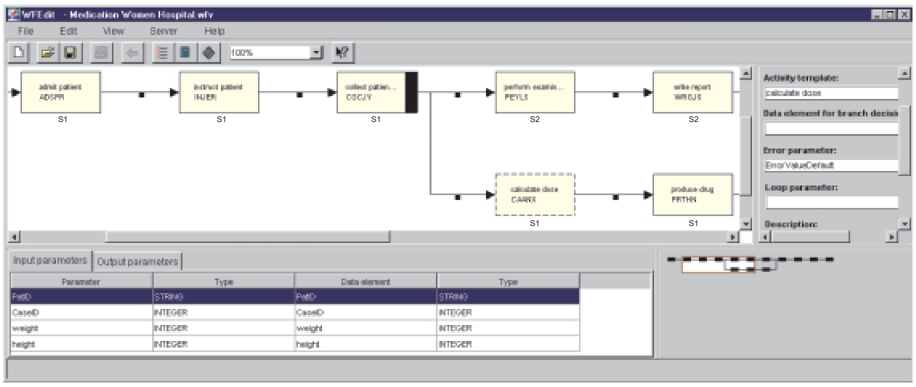
We have realized the described features in the ADEPT WfMS. This research prototype supports WF control and monitoring, demonstrates the feasibility of dynamic WF changes in a (distributed) WfMS, deals with temporal constraints, shows which user and programming interfaces are required, and proves that the concepts work in conjunction with each other as well. All system components have been implemented in Java, for communication Java RMI has been used.

3.1 ADEPT Buildtime Components

The ADEPT buildtime components enable the definition and management of WF templates, the description of inter-WF dependencies, the modeling of organizational entities, the specification of security constraints (Who is allowed to perform a particular WF change?), and the plug-in of application components. All relevant information is stored in the ADEPT repository. In addition, XML-based descriptions of model data may be generated; e.g., to export template descriptions to foreign tools or to exchange them between different WF servers. However, we do not support the XPDL syntax as defined by the Workflow Management Coalition (WfMC). On the one hand, the ADEPT WF meta model comprises several elements not captured by XPDL, on the other hand the support of WfMC standards does not have top priority in our research project.

For the modeling and management of WF templates, ADEPT offers a syntax-driven, graphical *WF editor*. A sample screen is depicted in Fig. 1. Its upper part shows a control flow window whereas the lower part displays input parameters of a selected activity and their mapping to WF data elements (data flow). Activity attributes are displayed in the right window. To each activity node a (reusable) template can be assigned. It sets out default properties like minimal/maximum duration, actor assignments (e.g., based on user roles), associated application components, and user-defined attributes. The WF designer is supported in correctly modeling and changing WF templates, i.e., static and dynamic WF properties as mentioned in Section 2 are guaranteed. To achieve this, the WF editor enables on-the-fly checks during WF editing as well as complete model checks initiated by the designer. In any case, a new WF template may only be released if all checks are successful. This is crucial for the WfMS to achieve a reliable and stable execution behavior. It is also a prerequisite for dynamic WF changes. Finally, new releases of a WF template are introduced by deploying the template to all relevant WF servers. For this, an XML-based description is sent to them and imported into their run-time databases.

ADEPT$_{distribution}$, the distributed variant of the ADEPT WfMS, additionally provides support for assigning WF servers to WF activities. This WF graph partitioning can be done manually or automatically by the use of a configuration tool. In the latter case, we make use of repository information (e.g., roles and locations of users) in order to determine optimal server assignments (i.e., to a find a partitioning which minimizes overall communication costs at run-time). Taking our example from Fig. 1, WF instances will be controlled by WF servers s_1 and s_2. (Server assignments are displayed below the activity nodes. Accordingly, “perform examination” and “write report” are controlled by s_2, whereas all other activities are carried out by s_1.)



Parameter	Type	Data element	Type
PatID	STRING	PatID	STRING
CaseID	INTEGER	CaseID	INTEGER
weight	INTEGER	weight	INTEGER
height	INTEGER	height	INTEGER

Fig. 1. ADEPT Workflow Editor

ADEPT provides several other buildtime components for defining different aspects of process-oriented information systems:

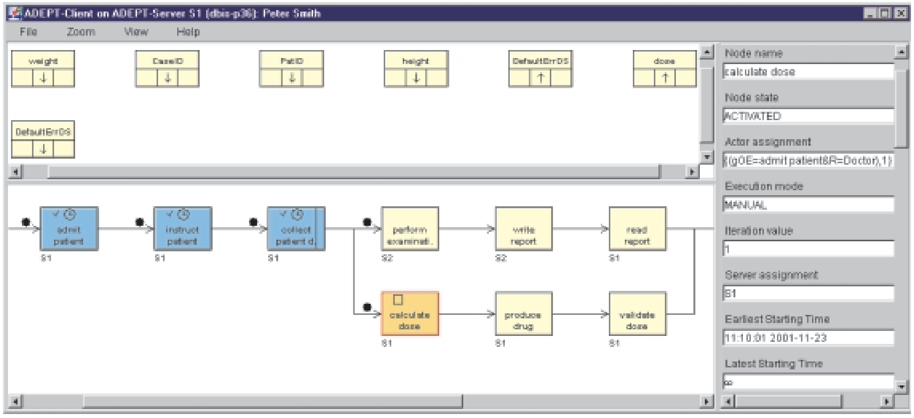
- **ADEPT interaction editor**: Powerful tool for defining and managing inter-workflow dependencies based on interaction expressions and graphs [4].
- **ADEPT organization modeler**: Graphical tool for describing organizational entities (e.g., user roles, capabilities, and organizational units) and their relationships (incl. substitution rules).
- **ADEPT application configuration tool**: This tool allows the WF designer to assign different application components to the same acitivity template. In doing so, the concrete binding of a component at runtime can be based on user as well as on workstation profiles.

3.2 ADEPT Runtime Clients

ADEPT comprises standard runtime clients for end users as well as for system and process administrators. These clients enable worklist display and manipulation, WF monitoring, activity program execution, dynamic WF changes, and system configuration.

For *worklist handling*, several client programs are available. Besides "thick" clients, ADEPT offers a Web client interface whose implementation is based on servlets. Web clients have a limited functionality when compared to standard WF clients, in particular concerning activity implementation. Both, thick and thin clients, however, already provide user interfaces for *dynamic changes*, giving end users the possibility, at run-time, to deviate from the pre-modeled task sequence. In detail, authorized actors may intervene into WF control by inserting, deleting, or shifting activities. In doing so, respective clients provide the necessary change context and allow change definition at a high semantic level. In particular, end users are not burdened with the complexity of dynamic changes; i.e., they must not deal with the problem of missing input data, the avoidance of deadlocks, or the graph transformations and state adaptations necessary to realize the change.

To monitor in-progress WF instances and to demonstrate the effects of dynamic changes, ADEPT offers a special *monitoring client*. It allows authorized users to visualize WF instance graphs, together with the information related to them. Fig. 2 shows a sample screen of a WF instance created from the template as depicted in Fig. 1. Activities "admit patient", "instruct patient", and "collect patient data" have been completed (indicated by symbol $\surd$), whereas activity "calculate dose" is currently activated (indicated by symbol $\Box$). Fig. 2 also displays data elements read and written by the selected activity ("calculate dose" in the example) as well as detailed information about this activity (e.g., actor and server assignments, starting time, priority, etc.). All relevant information is managed by the WF server which controls this activity (s_1 in the example). Actually, the monitoring client only shows the WF instance graph from the viewpoint of server s_1 (to which it is connected). Normally, this server does not know how far the execution in the upper branch of the parallel branching (currently controlled by s_2) has proceeded.



Node name
calculate dose

Node state
ACTIVATED

Actor assignment
{(gOE=admit patient&R=Doctor),1}

Execution mode
MANUAL

Iteration value
1

Server assignment
S1

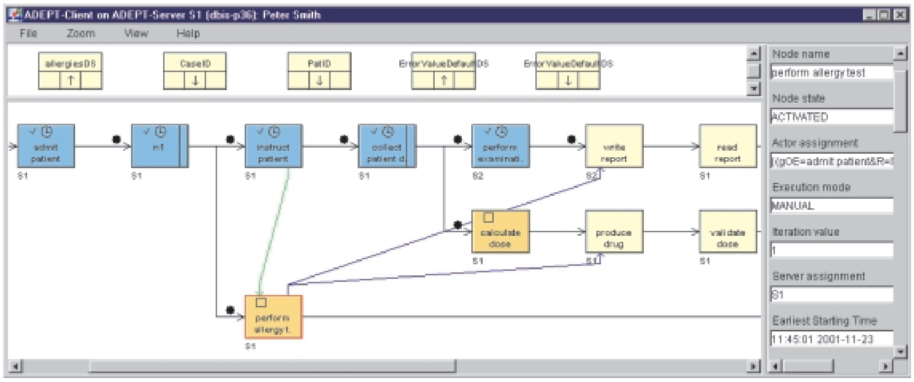
Earliest Starting Time
11:10:01 2001-11-23

Latest Starting Time
∞

Fig. 2. ADEPT Monitoring Client (before the dynamic change of a WF instance)

Let us sketch how a dynamic change of the (distributed) WF instance from Fig. 2 is realized:

Example: Assume that an authorized user (connected to s_1) specifies that activity "perform allergy test" is to be inserted between node sets {"instruct patient"} and {"write report", "produce drug"}; i.e., the allergy test shall be performed after patient instruction and before reporting and drug production. The resulting WF instance graph is depicted in Fig. 3. Internally, the change is accomplished as follows: First of all, to decide whether the insertion is permissible or not, s_1 retrieves information about the global state of the WF instance from other active servers (s_2 in our example). As a result, s_1 finds out that activities "write report" (controlled by s_2) and "produce drug" (controlled by s_1 itself) have not been started yet; thus the dynamic insertion is allowed. In the following, s_1 performs all necessary graph transformations to realize the change. It inserts activity "perform allergy test" parallel to the minimal block, which contains the nodes "instruct patient", "write report", and "produce drug". (For this, the AND split n_1, which represents a null task, is inserted). To enforce the desired control dependencies, three synchronization edges are added (e.g., the edge linking "perform allergy test" with "write report"). Finally, the state of the newly inserted activity is evaluated, leading to its immediate activation.



Node name
perform allergy test
Node state
ACTIVATED
Actor assignment
((gOE=admit patient&R=(
Execution mode
MANUAL
Iteration value
1
Server assignment
S1
Earliest Starting Time
11:45:01 2001-11-23

Fig. 3. ADEPT monitoring client (after the WF instance change)

3.3 ADEPT System Architecture and Programming Interfaces

The ADEPT WfMS is based on a multi-server architecture (cf. Fig. 4). A WF instance may either be controlled by a single server or by multiple servers if favorable. To each server different clients can be connected, e.g., worklist programs, monitoring components, and modeling tools. For implementing non-standard clients, ADEPT offers a rich API. It extends the one-directional client-server communication in order to enable WF servers to play an active role if need be; e.g., to initiate requests at the client site in order to get approvals from WF participants when performing a change or to immediately notify users when deadlines are going to be missed. Which communication model is used depends on the application scenario and can be configured by developers. Inter-WF dependencies are controlled by an interaction manager, which uses suitable coordination protocols to ensure that a client does not execute an action which is currently not permitted according to some inter-workflow dependency.

Server implementation is based on relational DBMS, which enables transactional execution of requests and, therefore, guarantees persistency and consistency of model and instance data. The kernel of the WF server is realized as a multi-layered architecture. The top level, the *Execution Layer*, processes client API calls (e.g., to start an activity or to perform a change). Each call is decomposed into a set of service requests from the underlying *Service Layer*, which comprises services designed along the described features (e.g., for scheduling WF activities, dynamically changing WF instances, managing user worklists,

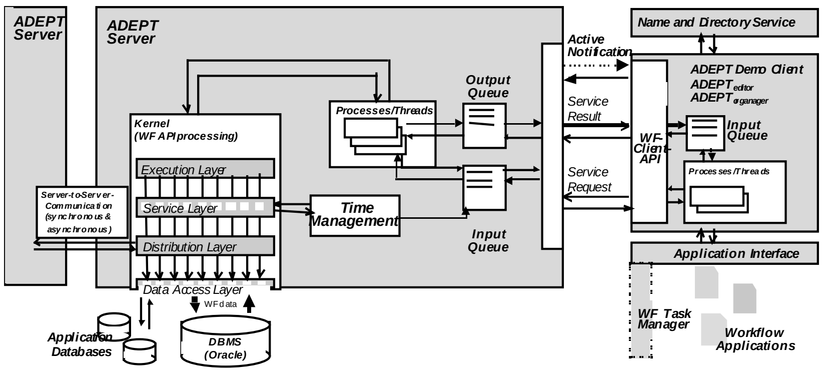


Fig. 4. ADEPT System Architecture

or handling temporal constraints). As an example take an activity completion, which leads to an update of the time schedule and the state of the respective WF instance, a role resolution of subsequent steps, and an update of worklists. Each component of the *Service Layer* itself decomposes calls into basic operations for the *Data Access Layer* (e.g., to read, to create, or to modify WF objects). Finally, if a migration of the WF control or a synchronization of the WF data becomes necessary, the *Distribution Layer* provides the required functionality.

ADEPT provides rich programming interfaces whose functionality goes far beyond the WfMC API. The offered change operations hide as much of the complexity of a dynamic change from application programmers as possible. Regarding activity insertion, for example, the method `dynamicInsertBetweenNodes` can be used: For a given WF instance, a new activity (with id `actIdentifier` and activity template `actTemplate`) can be inserted between node sets `predNodes` and `succNodes`. Information on how to map activity parameters to process data elements can be passed by the parameter `maInfo`. ADEPT allows different settings, e.g., automatic mapping of parameters to existing data elements or provision of input parameters by automatically generated, electronic forms.

```
public class WFProcessInstance {
      public WFModificationResult dynamicInsertBetweenNodes(
          ActivityTemplate actTemplate, ActivityId actIdentifier,
          InsertionArea predNodes, InsertionArea succNodes, ModificationAdjustInfo maInfo)
      // other methods
}
```

4 Practical Use and Lessons Learned

To gain concrete implementation and usability experience we have elaborated ADEPT within several research projects. Some of them have been carried out by our department in close cooperation with partners from different application domains. Additionally, we deployed the ADEPT WfMS to other research groups who have used it as platform for implementing sophisticated WF scenarios. In summary, all these projects helped us to identify basic needs for adaptive workflows and to evolve the ADEPT WfMS over time. Current projects working with ADEPT include CONSENSUS [1], AgentWork [3], and WebFlow [9].

Clinical workflows: We consider hospital processes as being one of the most challenging application areas for WF technology [2]. Typically, a hospital comprises decentralized units which participate in a variety of medical and organizational procedures with different complexity and duration (up to several months). In a two-years WF project with a Women's Hospital, we performed an in-depth analysis of all characteristic WF types, the organizational structures and responsibilities related to them, the kinds of exceptions which may occur, and the adequate reactions necessary to deal with them. The identified requirements helped us to design the basic features of the ADEPT WfMS and to get an impression of the change facilities needed. In order to gain concrete experience with the use of WF technology in general and with the ADEPT WfMS in particular we implemented selected clinical processes based on them. The goal was to learn how computer-based process support can be smoothly integrated in the daily routine work and how adequate user interfaces have to look like. As a result, it became obvious that WfMS with a proper, secure, and robust handling of exceptional cases are a mandatory prerequisite for any WF-based clinical application. ADEPT has been perfectly coherent with these requirements.

Automatic WF adaptations: AgentWork [3] offers a system for automatically adapting WF instances. For this, a rule-based approach has been applied. When exceptional events occur, AgentWork identifies WF instances to be adapted, determines the change operations to be applied, automatically performs the change, and notifies WF participants accordingly. AgentWork has adopted the ADEPT meta model and has used the ADEPT WfMS as implementation platform. It benefits from the offered features, in particular concerning WF modeling and execution, ad-hoc changes, and temporal constraint management. Apart from this, we had received important feedback which helped us to evolve the ADEPT user and programming interfaces. Currently, with WebFlow another project of this group is on its way [9]. It aims at the flexible support of cross-organizational workflows. Due to its dynamic change facilities, the ADEPT WfMS will be a core component of WebFlow as well.

Flexible E-negotiations: CONSENSUS offers a flexible support system for e-negotiations based on parameters like quality, delivery, or warranty [1]. E-negotiations are required, for example, in conjunction with supply chains and e-procurement. On the one hand they have to be organized in a process-oriented manner, on the other hand they require flexibility and dynamism to accommodate to the various contingencies and obstacles that can appear during negotiation. For example, if a supplier or a shipping company makes a new offer that might be of interest for a buying company, the buyer will review negotiation activities already planned within the WF model and may want to rearrange them (e.g., to dynamically skip, replace, or shift activities). In this context, the ADEPT change and verification facilities have proven as perfectly coherent with the flexibility requirements in e-negotiations. However, there are several requirements identified within the CONSENSUS project (e.g., dynamic change of WF attributes) which have not yet been fully supported by ADEPT (see [1]).

5 Summary

The ADEPT WfMS is the technological answer to the requirements set out by real-world processes. We have implemented fundamental concepts related to WF modeling, dynamic changes, temporal constraints, inter-WF dependencies, and scalability in a powerful research prototype. Currently, the integration of change propagation facilities in connection with WF schema evolution is on its way. The lessons learned from the sketched application projects have helped us to further develop the underlying concepts of the ADEPT WfMS, to improve and complement its buildtime and runtime components, and to refine user as well as programming interfaces. Adaptive WF technology as offered by ADEPT will be core of future WfMS and significantly influence the development of process-centered applications. It will drastically simplify application programming by providing rich, high-level interfaces for defining and changing model as well as instance data. As a consequence, development and adaptation times can be reduced by factors when compared to current "hard-wired" solutions.

References

1. Bassil, S., Benyoucef, M., Keller, R., Kropf, P.: Addressing dynamism in e-negotiations by workflow management systems. In: Proc. DEXA Workshop. (2002)
2. Dadam, P., Reichert, M., Kuhn, K.: Clinical workflows – the killer application for process-oriented information systems? In: Proc. 4th Int'l Conf. on Business Information Systems (BIS '00), Poznan, Poland (2000) 36–59
3. Müller, R., Rahm, E.: Dealing with logical failures for collaborating workflows. In: Proc. Int'l 5th Conf. on Coop. Inf. Sys., Eilat (2000) 210–223
4. Heinlein, C.: Workflow and process synchronization with interaction expressions and graphs. In: Proc. Int'l Conf. Data Eng., Heidelberg (2001) 243–252
5. Reichert, M., Dadam, P.: ADEPT$_{flex}$ - supporting dynamic changes of workflows without losing control. JIIS **10** (1998) 93–129
6. Reichert, M., Dadam, P., Bauer, T.: Dealing with backward and forward jumps in workflow management systems. Int'l Journal Software and Systems Modeling **2** (2003)
7. Bauer, T., Dadam, P.: Efficient distributed workflow management based on variable server assignments. In: Proc. CAiSE '00, Stockholm (2000) 94–109
8. Bauer, T., Reichert, M., Dadam, P.: Intra-subnet load balancing in distributed workflow management systems. Int'l Journal of Cooperative Information Systems (accepted for publication)
9. Greiner, U., Rahm, E.: WebFlow: A system for the flexible execution of web-based, cooperative workflows (in German). In: Proc. Database Systems For Business, Technology and Web (BTW'2003), Leipzig (2003)