Adaptive workflow

On the interplay between flexibility and support

W.M.P. van der Aalst[1,3] T. Basten[1] H.M.W. Verbeek[1] P.A.C. Verkoulen[1,2] M. Voorhoeve[1]
[1] *Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands, {wsinwa,tbasten,wsineric,verkoulen,wsinmarc}@win.tue.nl*
[2] *Origin DTS/Infrastructure Consultancy, Building VH-4; P.O. Box 6435, NL-5600 HK, Eindhoven, The Netherlands, peter.verkoulen@nl.origin-it.com*
[3] *Large Scale Distributed Systems, Department of Computer Science, University of Georgia, 415 Graduate Studies Research Center, Athens, GA 30602-7407, USA, wvdaalst@om.cs.uga.edu*

Key words: Workflow management, Workflow management systems, Process modeling, Petri nets, exception handling.

Abstract: Today's information systems do not support adaptive workflow: either the information system abstracts from the workflow processes at hand and focuses on the management of data and the execution of individual tasks via applications or the workflow is supported by the information system but it is hard to handle changes. This paper addresses this problem by classifying the types of changes. Based on this classification, issues such as syntactic/semantic correctness, case transfer, and management information are discussed. It turns out that the trade-off between flexibility and support raises challenging questions. Only some of these questions are answered in this paper; most of them require further research. Since the success of the next generation of workflow management systems depends on the ability to support adaptive workflow, it is important to provide answers for the questions raised in this paper.

1. INTRODUCTION

Recently, many Workflow Management Systems (WFMSs) have become available. These systems signify that the term 'workflow management' is not just another buzzword. The phenomenon workflow management will have a large impact on the next generation of information systems. As the workflow paradigm continues to infiltrate organisations that need to cope with complex administrative processes, the workflow management system will become a fundamental building block (Koulopoulos 1995; Jablonski & Bussler 1996). Therefore, the subject workflow management is of the utmost importance for people involved in the (re)design of administrative processes or the development of systems to support these processes.

At the moment, there are more than 200 workflow products commercially available (Lawrence 1997) and many organisations are introducing workflow technology to support their business processes. It is widely recognised that workflow management systems should provide *flexibility* (Van der Aalst et al 1998; Agostini & De Michelis 1998; Casati et al 1998; Ellis et al 1995; Ellis et al 1998; Han & Sheth 1998; Heinl et al 1998; Klein et al 1998; Voorhoeve & Van der Aalst 1996; Wolf & Reimer 1996). However, today's workflow management systems have problems dealing with *changes*. E.g., new technology, new laws, and new market requirements may lead to (structural) modifications of the workflow process definition at hand. In addition, ad-hoc changes may be necessary, e.g., because of exceptions. The inability to deal with various changes limits the application of today's workflow management systems.

Figure 1 shows the different fields of support for collaborative work. We distinguish between unstructured, information centric approaches (Computer-Supported Co-operative Work or CSCW) and structured, process centric ones (production workflow). Existing tools are typically in one of the two extremes of the spectre: groupware products such as Lotus Notes and Exchange are typical CSCW tools, not providing much process support, whereas commercially available (production) WFMSs such as Staffware and Flowmark are not able to cope with unstructuredness.

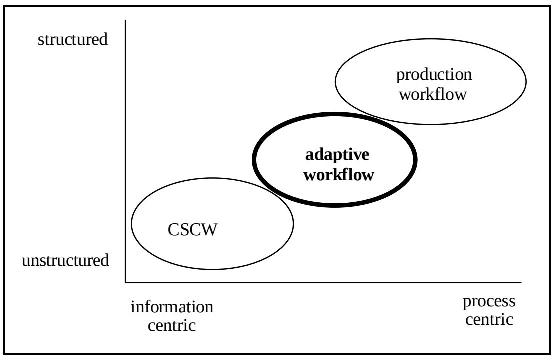

Figure 1. The workflow spectrum.

Linking production WFMSs to groupware products does not really solve the problem, as the process logic is then still handled by the same inflexible workflow engine. To bridge the gap between CSCW and production workflow, several research groups are working on the problems associated to *adaptive workflow*.

Adaptive workflow aims at providing process support like normal workflow systems do, but in such a way that the system is able to deal with certain changes. These changes may range from ad-hoc changes such as changing the order of two tasks for an individual case (often called *exceptions*) to the redesign of a workflow process as the result of a Business Process Redesign (BPR) project.

Typical issues related to adaptive workflow are:

- *Correctness*
  What kind of changes are allowed and is the resulting workflow process definition correct with respect to the criteria specified? We distinguish syntactic correctness (e.g. is it still a workflow?) and semantic correctness (e.g. can existing cases in the system be finished in a proper way?).
- *Dynamic change*
  What to do with running instances (cases) of a workflow of which the definition has been changed? The term dynamic change refers to the problems that occur when running cases have to migrate from one process definition to another.
- *Management information*
  How to provide a manager with aggregated information about the actual state of the workflow processes?

Taking these issues into account, a classification of the types of changes is presented. Based on this classification, potential problems are identified and pointers to solutions based on Petri-net theory are given.

2. ESSENCE OF WORKFLOW MODELING

The term *Workflow Management* actually refers to the “logistics” of business processes. Workflow management does not focus on what information is being passed in a business process, but more on the control of the activity chain that is necessary to execute the business processes. The Workflow Management Coalition (WfMC) defines a workflow management system as follows: “*A system that completely defines, manages, and executes workflows through the execution of software whose order of execution is driven by a computer representation of the workflow logic*” (Lawrence 1997). Workflows are *case-based*, i.e., every piece of work is executed for a specific case: an order, an insurance claim, a tax declaration, etc. The objective of a workflow management system is to handle these cases (by executing *tasks*) as efficiently and effectively as possible. The *workflow process definition* specifies which tasks need to be executed and in what order. When a task is executed for a case, this is usually done by using one or more *resources*, e.g., a machine, an employee, etc.

As can be seen in *Figure 2*, we identify three dimensions of a workflow (Van der Aalst 1998b):

i. the *case dimension*, making clear that all cases are dealt with independently,
ii. the *process dimension*, specifying the overall workflow process, abstracting from individual cases,
iii. the *resource dimension*, depicting and classifying the resources used in the process.

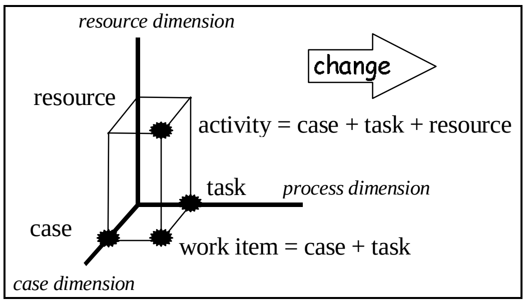

Figure 2. Three dimensions of workflow.

In our work, we use Petri nets to represent workflow systems (Van der Aalst 1998a; Van der Aalst 1998b; Ellis & Nutt 1993). A Petri net represents a workflow iff it has exactly one starting place (*source*) and exactly one end place (*sink*) and the net obtained by adding a transition with the sink as the only input place and the source as the only output place is strongly connected. The latter is the case if for every two nodes *x* and *y* in the net, there is a path from *x* to *y*. Petri nets with this particular structure are called *workflow nets*. *Figure 6* shows two workflow nets. The left-hand-side process describes the sequential execution of four tasks (*A*, *B*, *C*, and *D*). Each task is modelled by a transition. Tasks are connected by places (represented by circles) to specify the ordering of tasks. Places may contain tokens (represented by black dots). A transition, i.e., a task, is enabled if and only if each of the input places contains a token. Enabled transitions can fire while removing tokens from the input places and putting tokens on the output places. In the sequential process shown in *Figure 6*, transition *C* is enabled because there is a token in the input place. Firing *C* will result in a situation where *D* is enabled. In the right-hand-side process in *Figure 6* tasks *B* and *C* are in parallel, i.e., they can be executed in any order. Note that *A* consumes one token and produces two tokens and that *D* consumes two tokens and produces one token. A detailed description of the class of workflow nets is beyond the scope of this paper and not needed for the remainder. However, some basic knowledge of Petri nets is needed to fully understand the concepts.

3. CLASSIFICATION OF CHANGE

This section deals with the different kinds of changes and their consequences. Some of the perspectives relevant for change are:

- *process perspective*, e.g., tasks are added or deleted or their ordering is changed,
- *resource perspective*, e.g., resources are classified in a different way or new classes are introduced,
- *control perspective*, e.g., changing the way resources are allocated to processes and tasks,
- *task perspective*, e.g., upgrading or downgrading tasks,
- *system perspective*, e.g., changes to the infrastructure or the configuration of the engines in the enactment service.

In this paper, we restrict ourselves to changes in the process perspective as indicated in *Figure 2*.

Figure 3 shows that two kinds of change are identified:

- *individual (ad-hoc) changes*
  Ad-hoc adaptation of the workflow process: a single case (or a limited set of cases) is affected. A good example is that of a hospital: if someone enters the hospital with a cardiac arrest, you are not going to ask him for his ID, although the workflow process may prescribe this. *Figure 3* distinguishes entry time changes (changes that occur when a case is not yet in the system) and on-the-fly changes (while in the system, the process definition for a case changes).
- *structural (evolutionary) changes*
  Evolution of the workflow process: all new cases benefit from the adaptation. A structural change is typically the result of a BPR effort. An example of such a change is the change of a 4-year curriculum at a university to a 5-year one.

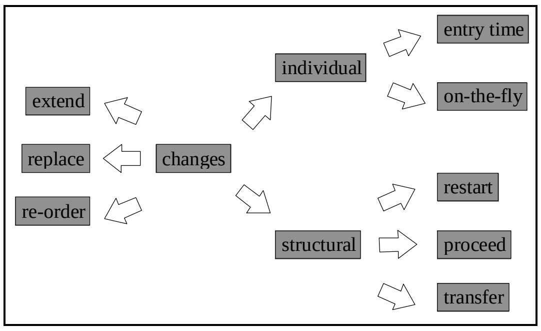

Figure 3. Classification of change.

At the left-hand side in *Figure 3*, we see the three different ways in which a workflow can be changed:

- the process definition is *extended* (e.g. by adding new tasks to cover process extensions),
- tasks are *replaced* by other tasks (e.g. a task is refined into a subprocess),
- tasks in the process are *re-ordered* (e.g. two sequential tasks are put in parallel).

Figure 3 gives three possible alternatives for handling existing cases in the system when a process definition changes. Dealing with existing cases is only relevant in the case of a structural change because individual changes will always be (similar to) exceptions and as such will be dealt with by the one who initiated the change explicitly.

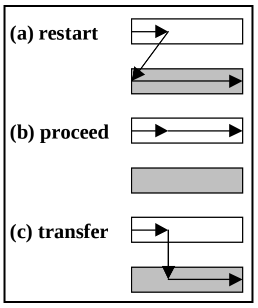

Figure 4. How to handle running cases?

Figure 4 shows the three alternatives: (a) *restart*: running cases are rolled back and restarted at the beginning of the new process, (b) *proceed*: changes do not affect running cases by allowing for multiple versions of the process, and (c) *transfer*: a case is transferred to the new process. The term *dynamic change* (Ellis et al 1995) is used to refer to the latter policy.

4. CORRECTNESS

In the previous section, we presented some of the possibilities to change workflow specifications. In addition, we have presented some alternatives for dealing with cases that are somewhere in their workflow life cycle the moment that the workflow is being changed.

These changes actually only make sense when they can be performed such that we can guarantee beforehand that the transformation satisfies a certain set of correctness-preservation properties. If we would not be able to make any statements about correctness preservation, it would mean that the new system would have no relationship with the old one. Being interested in adaptive workflow, this is obviously an undesired situation. Thus, it is very important that verification techniques and supporting tools are being developed that support such correctness-preserving transformations.

Two different kinds of correctness notions can be distinguished, viz. syntactic and semantic correctness.

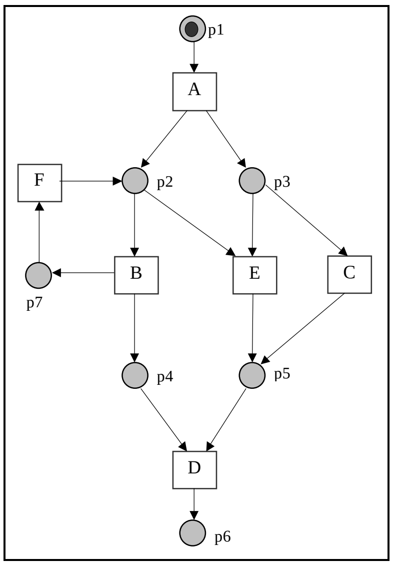

Figure 5. An incorrect workflow process definition.

S*yntactic* correctness is independent of the context, i.e., it refers to the minimal requirements any workflow should satisfy. For example, there should be no tasks without input places. Note that syntactic correctness not only refers to the structure of the workflow but also to the dynamic behaviour, e.g., potential deadlocks and livelocks are not allowed. A very important correctness criterion is the so-called “soundness property” that guarantees proper termination (Van der Aalst 1998b). Proper termination means that the state is reached with a single token in the sink place. *Figure 5* shows an incorrect workflow process definition. The execution of task *E* before task *B* will result in a deadlock because the case gets stuck in the state with just a token in *p5*. If task *C* is executed, then it is possible to execute *D* but a livelock is created because it is not possible to escape from the cycle formed by *B* and *F*. The workflow can be corrected by adding an arrow from *E* to *p2*, removing *p7*, and adding an arrow from *p4* to *F*. Today, Petri-net theory provides adequate tools to verify syntactic correctness. Nevertheless, such facilities are still missing in most of the commercial workflow management systems.

Semantic correctness is concerned with the context in which the change occurs. Intuitively, semantic correctness deals with similarities between the capabilities of the old workflow and the capabilities of the new workflow. E.g., it may be desirable that the new workflow is able to handle cases the old way (but probably has some more functionality). Consider for example the two process definition shown in *Figure 6*: the right-hand process can do more than the left-hand process but not vice versa. A similar relation holds between the right-hand process in *Figure 6* and the corrected version of the process shown in *Figure 5*. Comparing different variants of the same workflow requires advanced inheritance concepts (Basten 1998; Van der Aalst & Basten 1997; Voorhoeve & Van der Aalst 1996). Research efforts should be aiming at defining semantic correctness and providing easy-to-use verification methods.

5. DYNAMIC CHANGE

We identified three ways to deal with running cases after a structural process change: (a) restart, (b) proceed, and (c) transfer. Restarting cases causes no real difficulties except that it is often difficult to rollback the tasks that have already been executed. The proceed policy causes no problems. In fact, it is the only policy truly supported by today’s commercial workflow management systems. The only policy that causes serious theoretical *and* practical problems is the transfer of cases. The term dynamic change refers to the problem of handling old cases in a new process, e.g., how to transfer cases to a new, i.e., improved, version of the process.

Figure 6 illustrates the dynamic change problem. If the sequential workflow process (left) is changed into the workflow process where tasks *B* and *C* can be executed in parallel (right) there are no problems, i.e., it is always possible to transfer a case from the left to the right. The sequential process has five possible states and each of these states corresponds to a state in the parallel process. For example, the state with a token in *s3* is mapped onto the state with a token in *p3* and *p4*. In both cases, tasks *A* and *B* have been executed and *C* and *D* still need to be executed.

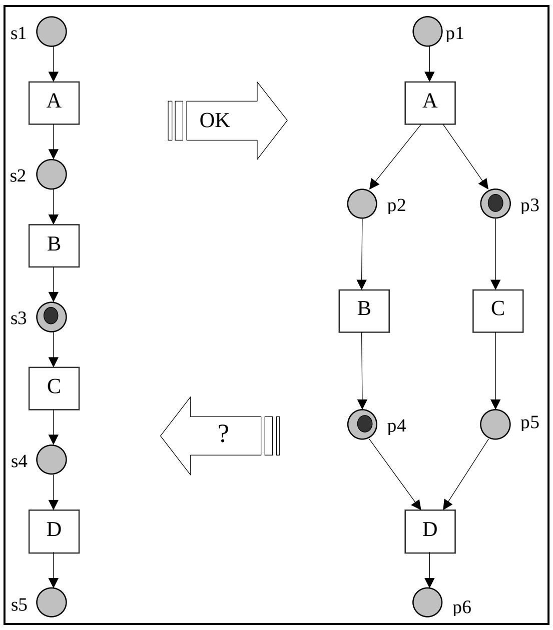

Figure 6. The dynamic change problem.

Now consider the situation where the parallel process is changed into the sequential one, i.e., a case is moved from the right-hand-side process to the left-hand-side process. For most of the states of the right-hand-side process this is no problem, e.g., a token in *p1* is moved to *s1*, a token in *p3* and a token in *p4* are mapped onto a single token in *s3*, and a token in *p4* and a token in *p5* are mapped onto a single token in *s4*. However, the state with a token in *p2* and *p5* (*A* and *C* have been executed) causes problems because there is no corresponding state in the sequential process (it is not possible to execute *C* before *B*). This simple example shows that it is not straightforward to migrate old cases to the new process after a change. Some authors have proposed a solution for the dynamic change problem (Agostini & De Michelis 1998; Ellis et al 1995; Ellis et al 1998). However, these solutions either require human intervention or are restricted to workflows with a particular structure. Note that both changes in *Figure 6* correspond to the reordering of tasks (see classification in Section 3).

6. MANAGEMENT INFO

Another problem of change is that it typically leads to multiple variants of the same process. For evolutionary (i.e. structural) change the number of variants is limited. Ad-hoc changes may lead to the situation where the number of variants may be of the same order of magnitude as the number of cases. To manage a workflow process with different variants it is desirable to have an aggregated view of the work in progress. Note that in a manufacturing process the manager can get a good impression of the work in progress by walking through the factory. For a workflow process handling digitised information, this is not possible. Therefore, it is of the utmost importance to supply the manager with tools to obtain a condensed but accurate view of the workflow processes. *Figure 7* shows a workflow processes with two variants: a sequential one (left) and a parallel one (middle). The numbers in the places indicate the number of cases in a specific state, e.g., in the sequential process there are 3 cases in-between task *B* and task *C*, and in the parallel process there are 2 cases in-between *A* and *B*. Since the manager requires an aggregated view rather than a view for every variant of the workflow process, the cases need to be mapped onto a generalised version of the different processes. A solution is to find the 'greatest common divisor' (gcd) or the 'least common multiple' (lcm) for the two processes shown. Since all the states of the sequential process can be represented in the parallel process, we choose the parallel process to present management information. *Figure 7* shows the aggregated view of the two workflow processes (right). For all places in the right-hand-side process except *m3*, it is quite straightforward to verify that the numbers are correct. The number of tokens in place *m3* corresponds to the number of cases in-between *A* and *C*. In the sequential process, there are 1+3=4 cases in-between *A* and *C*. In the parallel process, there are also 4 cases in-between *A* and *C*, which brings the total to 8. For this small example, it may seem trivial to obtain this information. However, in general there are many variants of processes which may have up to 100 tasks and it is far from trivial to present aggregated information to the manager. The topic of generating management information was addressed in (Voorhoeve & Van der Aalst 1996; Voorhoeve & Van der Aalst 1997). Despite its relevance for the next generation of workflow management systems only few researchers seem to be working on this topic. At the moment, only intuitive notions for gcd and lcm exist.

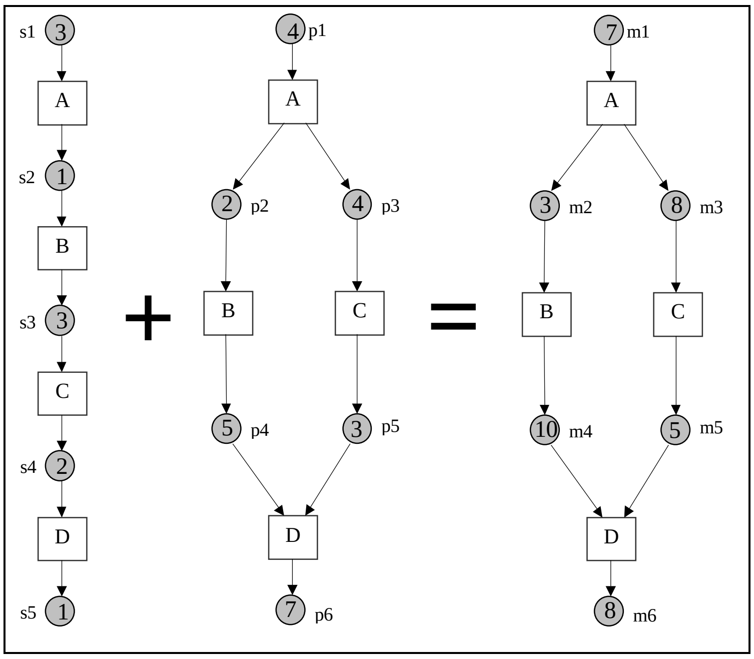

Figure 7. Mapping cases in different processes onto the "least common multiple" of all processes.

7. CONCLUSION

In this paper, we discussed some of the problems that need to be solved to enable adaptive workflow. Most of the problems stem from the fact that flexibility (the ability to handle changes) on the one hand and process support (enactment, control, and management information) on the other hand impose (partially) conflicting constraints. We have classified the various forms of change. Based on this classification we pointed out some solutions for the problems identified. Future work in this area will focus on verification (semantic correctness), relating different versions of a workflow process for supporting dynamic change, and generating management information. It is our belief that the inheritance-preserving transformation rules presented in (Van der Aalst & Basten 1997; Basten 1998) are a good starting point for solving the problems identified. Each of the rules corresponds to a design construct which is often used in practice, namely choice, parallel composition, sequential composition, and iteration. The rules preserve to some extent syntactic and semantic correctness. Future research will be aimed at exploiting the transformations to deal with dynamic change and the generation of useful management information.

8. REFERENCES

Aalst, W.M.P. van der 1998a. Chapter 10: Three Good reasons for Using a Petri-net-based Workflow Management System. In T. Wakayama et al., editors, Information and Process Integration in Enterprises: Rethinking documents, Kluwer Academic Publishers.

Aalst, W.M.P. van der 1998b. The Application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers, 8(1):21-66.

Aalst, W.M.P. van der & Basten, T. 1997. Life-cycle Inheritance: A Petri-net-based approach. In P. Azéma and G. Balbo, editors, Application and Theory of Petri Nets 1997, volume 1248 of Lecture Notes in Computer Science, pages 62-81. Springer, Berlin, Germany.

Aalst, W.M.P. van der & Michelis, G. De & Ellis, C.A. (editors) 1998. Proceedings of Workflow Management: Net-based Concepts, Models, Techniques and Tools (WFM’98). Computing Science Report 98/7, Eindhoven University of Technology, Eindhoven, the Netherlands.

Agostini, A. & De Michelis, G. 1998. Simple Workflow Models. In (Van der Aalst & De Michelis & Ellis 1998), pages 146-164.

Basten, T. 1998. In Terms of Nets: System Design with Petri Nets and Process Algebra. PhD Thesis. Eindhoven University of Technology, Department of Computing Science, Eindhoven, the Netherlands.

Casati, F. & Ceri, C. & Pernici, B. & Pozzi, G. 1998. Workflow Evolution. Data and Knowledge Engineering, 24(3):211-238.

Ellis, C.A. & Keddara, K. & Rozenberg, G. 1995. Dynamic change within workflow systems. In N. Comstock and C.A. Ellis, editors, Conf. on Organizational Computing Systems, pages 10 - 21. ACM SIGOIS, ACM. Milpitas, California.

Ellis, C.A. & Keddara, K. & Wainer, J. 1998. Modeling Workflow Dynamic Changes Using Timed Hybrid Flow Nets. In (Van der Aalst & De Michelis & Ellis 1998), pages 109-128.

Ellis, C.A. & Nutt, G.J. 1993. Modelling and Enactment of Workflow Systems. In M. Ajmone Marsan, editor, Application and Theory of Petri Nets 1993, volume 691 of Lecture Notes in Computer Science, pages 1-16. Springer, Berlin, Germany.

Han, Y. & Sheth, A. 1998. On Adaptive Workflow Modeling. In Proceedings of the 4th International Conference on Information Systems Analysis and Synthesis, pages 108-116, Orlando, Florida.

Heinl, P. & Horn, S. & Jablonski, S. & Neeb, J. & Stein, K. & Teschke, M. 1998. A comprehensive approach to flexibility in workflow management systems. Technical report TR-16-1998-6, University of Erlangen-Nuremberg, Erlangen, Germany.

Jablonski, S. & Bussler, C. 1996. Workflow Management: Modeling Concepts, Architecture, and Implementation International Thomson Computer Press.

Klein, M. & Dellarocas, C. & Bernstein, A. (editors) 1998. Proceedings of the CSCW-98 Workshop Towards Adaptive Workflow Systems, Seattle.

Koulopoulos, T.M. 1995. The Workflow Imperative. Van Nostrand Reinhold, New York.

Lawrence, P. (editor) 1997. Workflow Handbook 1997, Workflow Management Coalition. John Wiley and Sons, New York.

Sheth, A. 1997. From Contemporary Workflow Process Automation to Dynamic Work Activity Coordination and Collaboration. Siggroup Bulletin, 18(3):17-20.

Voorhoeve, M. & Aalst, W.M.P. van der 1996. Conservative Adaption of Workflow. In (Wolf & Reimer 1996), pages 1-15.

Voorhoeve, M. & Aalst, W.M.P. van der 1997. Ad-hoc Workflow: Problems and Solutions. In R. Wagner, editor, Proceedings of the 8th DEXA Conference on Database and Expert Systems Applications, pages 36-41, Toulouse, France.

Wolf, M. & Reimer, U. (editors) 1996. Proceedings of the International Conference on Practical Aspects of Knowledge Management (PAKM’96), Workshop on Adaptive Workflow, Basel, Switzerland.