Woflan: A Petri-net-based Workflow Analyzer

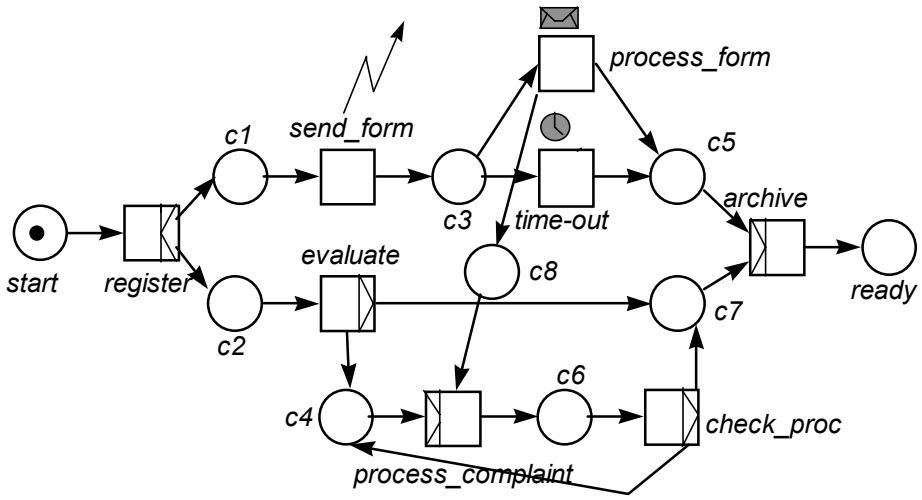
W.M.P. van der Aalst
Department of Mathematics and Computing Science,
Eindhoven University of Technology,
P.O. Box 513, NL-5600 MB, Eindhoven,
The Netherlands, telephone: -31 40 2474295,
e-mail: wsinwa@win.tue.nl

Abstract

Workflow management promises a new solution to an age-old problem: controlling, monitoring, optimizing and supporting business processes. What is new about workflow management is the explicit representation of the business process logic which allows for computerized support. Unfortunately, today's systems give hardly any support to verify the correctness of workflow processes. This paper discusses a verification tool *Woflan*. Woflan uses state-of-the-art results from Petri net theory, and interfaces with some of the leading workflow tools at the Dutch market. This paper describes the architecture of Woflan and illustrates its functionality by means of some small examples.

1. Introduction

This paper describes *Woflan*, a tool which analyzes workflow process definitions specified in terms of Petri nets. Woflan (WOrkFLow ANalyzer) has been designed to verify process definitions which are downloaded from a workflow management system (cf. [7,14]). Clearly, there is a need for such a verification tool, because today's workflow management systems do not support advanced techniques to verify the correctness of workflow process definitions. These systems typically restrict themselves to a number of (trivial) syntactical checks. Therefore, serious errors such as deadlocks and livelocks may remain undetected. This means that an erroneous workflow may go into production, thus causing dramatic problems for the organization. An erroneous workflow may lead to extra work, legal problems, angry customers, managerial problems, and depressed employees. Therefore, it is important to verify the correctness of a workflow process definition before it becomes operational.



• Figure 1: An erroneous workflow process definition.

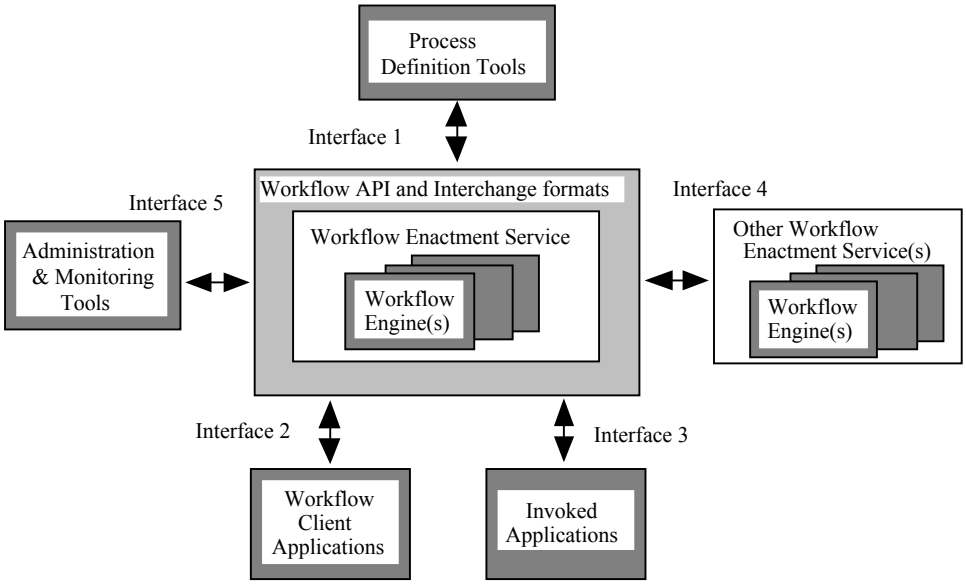
Consider for example the workflow process definition shown in Figure 1. The process handles complaints which enter the system via the place *start* and leave via the place *ready*. First the complaint is registered by executing the task *register*, then in parallel a questionnaire is sent to the complainant (task *send_form*) and the complaint is evaluated (task *evaluate*). If the complainant returns the questionnaire within two weeks, the task *process_form* is executed. If the questionnaire is not returned within two weeks, the result of the questionnaire is discarded (task *time_out*). Based on the result of the evaluation, the complaint is processed or not. The actual processing of the complaint (task *process_complaint*) is delayed until the form has been processed. The processing of the complaint is checked via task *check_proc*. Finally, task *archive* is executed. In Figure 1, the workflow process definition is specified in terms of a Petri net. The tasks *register*, *send_form*, *evaluate*, *process_form*, *time_out*, *process_complaint*, *check_proc* and *archive* have been modeled by transitions. To model the states between tasks, conditions have been added. Each condition is modeled by a place. For example, place *c2* corresponds to the condition `ready to evaluate complaint'. Condition *c5* is true (i.e. place *c5* contains a token) if the questionnaire has been processed or a time-out has occurred. Transition *register* is a so-called AND-split, i.e., a token is produced for each of its output places. The transitions *evaluate* and *check_proc* are OR-splits, i.e., precisely one of the output places obtains a token.

A close observation of the workflow process definition shown in Figure 1 reveals several design errors. If the complaint needs to be processed and the complainant does not return the form in time, the complaint will deadlock with a token in *c4* and *c5*. A similar deadlock occurs if the complaint needs to be processed for a second time. Moreover, if the complaint does not need to be processed, place c8 may contain a dangling reference after archiving the complaint. Although the workflow process definition shown in Figure 1 may have dramatic consequences, it can be designed and made operational using almost all of the workflow management systems available today. These systems will not warn for the potential deadlock. We have developed Woflan to assist the workflow designer to detect and repair errors such as the one shown in Figure 1.

This paper is organized as follows. First, we discuss the role of a workflow verification tool in the context of the standard architecture for workflow management systems. Then we discuss the architecture and functionality of Woflan in more detail. We also give pointers to two workflow tools which can interface with Woflan: COSA (COSA Solutions/Software-Ley, Pullheim, Germany, [19]) and Protos (Pallas Athena, Plasmolen, The Netherlands, [18]).

2. The need for a workflow verification tool

In the last five years, *workflow management systems* [15, 20] have become a popular tool to support the logistics of business processes in banks, insurance companies, and governmental institutions. Until recently there were no generic tools to support workflow management. As a result, parts of the business process were hard-coded in the applications. For example, an application to support task X triggers another application to support task Y. This means that one application knows about the existence of another application. This is undesirable, because every time the underlying business process is changed, applications need to be modified. Moreover, similar constructs need to be implemented in several applications and it is not possible to monitor and control the entire workflow. Therefore, several software vendors recognized the need for workflow management systems. A workflow management system (WFMS) is a generic software tool which allows for the definition, execution, registration and control of workflows. At the moment, many vendors are offering a workflow management system. This shows that the software industry recognizes the potential of workflow management tools.



• Figure 2: The reference architecture of the WfMC (© WfMC).

In this paper we restrict ourselves to workflow management systems which support the processing of large amounts of cases according to predefined process definitions expressed in some formal language. The *Workflow Management Coalition* (WfMC) also focuses on this type of software tools (cf. [15,20]). Many vendors and users of these workflow management systems have joined the WfMC to identify the common characteristics of these tools, to standardize terminology and define standard architectures and interfaces. One of the first results achieved by the WfMC was the definition of a reference model. Figure 2 gives an overview of this reference model. The reference model describes the major components and interfaces within a workflow architecture. The core of any workflow system is the *workflow enactment service*. The workflow enactment service provides the run-time environment which takes care of the control and execution of the workflow. For technical (scalability) and/or managerial reasons the workflow enactment service may use multiple *workflow engines*. Each workflow engine handles selected parts of the workflow and manages selected parts of the resources. *The process definition tools* are used to specify and analyze workflow process definitions and/or resource classifications. These tools are used at design time. In most cases, the process definition tools can also be used as a BPR-toolset. Most workflow management systems provide three process definition tools: (1) a graphical interface to define workflow processes, (2) an interface to specify resource classes (organizational model) and (3) a simulation tool to analyze a specified workflow. The end-user communicates with the workflow system via the *workflow client applications*. An example of a workflow client application is the well-known *in-basket*. Via such an in-basket work items are offered to the end user. By selecting a work item, the user can execute a task for a specific case. If necessary, the workflow engine invokes applications via interface 3. The *administration and monitoring tools* are used to monitor and control the workflow. These tools are used to register the progress of cases and to detect bottlenecks. Moreover, these tools are used to set parameters, allocate people and handle abnormalities. Via interface 4 the workflow system can be connected to other workflow systems. The architecture shown in Figure 2 has been adopted by most vendors. Unfortunately, only interface 2 has been standardized.

By using a workflow management system, it becomes easy to modify an existing workflow or to design a new one. This way, the system allows for the realization of flexible information

systems. However, there is a potential problem: the (re)designed workflow may contain dramatic errors. These errors may result in deadlocks, lost cases, livelocks, and dangling references. Therefore, it is important to verify the correctness of the workflow process definition before it becomes operational. Today's workflow management systems support some checks. Most of these checks are at a syntactical level and can only be used to detect trivial mistakes. Today's workflow management are weak with respect to the verification of workflow process definitions, because they often use an ad-hoc process modeling technique (i.e. they have to invent their own verification techniques), the verification techniques are difficult to implement, and the results are often difficult to interpret. To aid the user in verifying a workflow process definition, we have developed Woflan. Woflan detects potential errors such as deadlocks, lost cases, livelocks, and dangling references at design time. In the architecture of the WfMC, Woflan is part of the process definition tools. Ideally, Woflan should be able to use interface 1. Unfortunately, interface 1 has not been standardized yet and the proposed standard for this interface is at a technical level with the emphasis on syntax instead of semantics. There is no consensus at a conceptual level. In the proposed standard, the syntax of the exchange format has been defined without formalizing the meaning of states and essential building blocks such as the AND/OR-split/join. Therefore, we have two choose between two approaches: (1) to integrate Woflan in a specific workflow management system, or (2) to use a conceptual standard which clearly specifies the semantics of the basic constructs needed. We have chosen for a mixture of these two approaches. Woflan is based on a conceptual standard but can interface with a limited number of workflow tools. The conceptual standard uses Petri nets (cf. [17]) as a starting point. Petri nets are a well-founded process modeling technique and allow for a graphical representation which is close to the representation in many workflow management systems (cf. [1,3,11,12,16,21]). Moreover, many analysis techniques have been developed for Petri nets (cf. [5,10,17]). These techniques have been put to work in Woflan.

3. Architecture of Woflan

As indicated in the introduction, Woflan (WOrkFLow ANalyzer) is a Petri-net-based tool to analyze the correctness of a workflow. Woflan has been developed by members of the SMIS group within the Department of Mathematics and Computing Science of Eindhoven University of Technology. The tool is workflow management system independent, i.e., a number of import functions to download workflow-scripts in Woflan (e.g. from COSA and Protos) are provided. Woflan uses standard Petri-net-based analysis techniques. However, the analysis results are presented in such a manner that end-users can understand the output of Woflan. Moreover, Woflan guides the end-user in correcting an erroneous workflow.

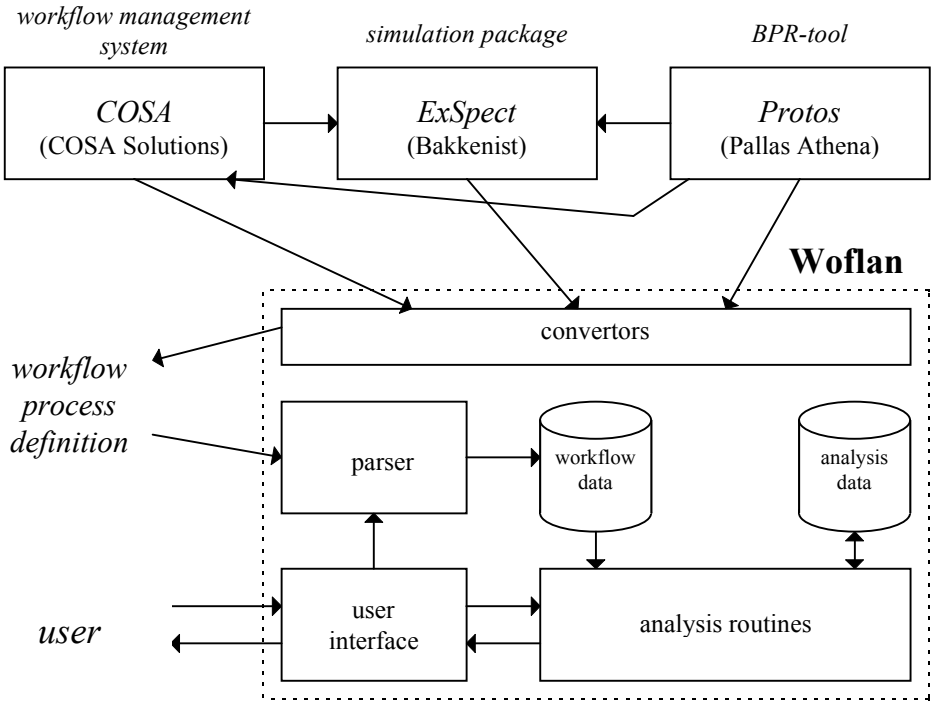
Woflan consists of three main parts:

- *parser*
  Woflan can analyze workflow process definitions specified in terms of a Petri net. The Petri net is assumed to have a special structure: there is one input place denoting the entry point of the workflow and one output place denoting the exit point of the workflow. Moreover, every node in the Petri net should be on a path from the input place to the output place. The parser reads the Petri net from an input file and builds up a data structure for each workflow process definition that needs to be analyzed. If the Petri net does not satisfy the requirements just mentioned, Woflan will warn the user.
- *analysis routines*
  The data structure created by the parser is used as a starting point for all kinds of analysis. On command, Woflan will provide the user with information about the structure of the workflow process definition. For example, Woflan will warn for tasks without any input or output condition, and detect potential errors by listing suspicious constructs, e.g., constructs violating the free-choice property, AND-split's complemented by OR-join's,

OR-split's complemented by AND-join's (well-structuredness), and parts of the net which are not S-coverable. Woflan will also detect dynamic errors by listing unbounded places, non-safe places, dead transitions and non-live transitions. For more advanced users, Woflan can generate place and transition invariants. The absence or presence of certain invariants may indicate the source of an error. Finally, Woflan will verify the soundness property ([5]). Soundness is one of the key concepts Woflan is based on. Therefore, we will elaborate on the soundness property in the next section.

- *user interface*
  To present the analysis results to the user, Woflan is equipped with a graphical user interface. The user interface has been built by using XVT. XVT is a software package to build user interfaces which are portable. In Woflan, multiple workflow process definitions can be analyzed at the same time. Each workflow process definition corresponds to a separate window in the user interface. By pushing buttons the user can focus on certain aspects of the process definition.

Figure 3 shows the architecture of Woflan.



• Figure 3: The architecture of Wolfan and the workflow tools it can interface with.

In addition to the parser, the analysis routines and the user interface, there will be a module for each WFMS which can interface with Woflan. At the moment there is one such module which can convert COSA script files into the format used by Woflan. COSA (COSA Solutions) is one of the leading products at the Dutch workflow market. COSA allows for the modeling and enactment of complex workflow processes which use advanced routing constructs. However, COSA does not support verification. Fortunately, Woflan can analyze any workflow process definition constructed by using CONE (COSA Network Editor), the design tool of the COSA system. Woflan can also import process definitions made with Protos. Protos (Pallas Athena) is a so-called BPR-tool. Protos supports Business Process Reengineering (BPR) efforts and can be used to model and analyze business processes. The tool is very easy to use and is based on Petri nets. To facilitate the modeling of simple workflows by users not familiar with Petri nets, it is possible to abstract from states. However, Protos cannot detect subtle design flaws which may result in deadlocks or livelocks. Therefore, it is useful to download workflows specified with Protos and analyze

them with Woflan. It is also possible to import process models made with ExSpect. ExSpect is a general purpose simulation tool based on high-level Petri nets. Moreover, workflow process definitions made with Protos or COSA can be simulated with ExSpect. Figure 3, also shows the relation between Woflan, COSA, Protos and ExSpect.

The current version of Woflan runs under Solaris (PC and UNIX workstations), Windows 3.11 and Windows 95 (PC). The system requirements are limited. The minimal configuration under Windows 95 is a Pentium PC with 8MB memory and 4MB free disk space).

4. Analysis techniques supported by Woflan

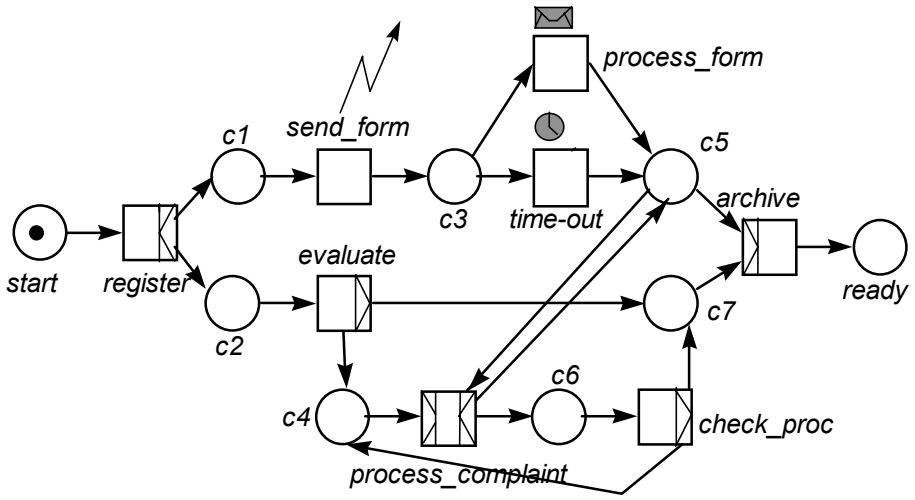
Two of the key concepts Woflan is based on, are the definition of a workflow net and the soundness property. For a formal definition of these two key concepts the reader is referred to [5]. The purpose of this paper is to illustrate the functionality of Woflan, not to review state-of-the-art results in Petri-net theory.

A Petri net which models a workflow process definition (i.e. the life-cycle of one case in isolation) is called a *workflow net* (WF-net). A workflow net satisfies two requirements. First of all, a workflow net has one input place (*i*) and one output place (*o*). A token in *i* corresponds to a case which needs to be handled, a token in *o* corresponds to a case which has been handled. Secondly, in a workflow net there are no dangling tasks and/or conditions. Every task (transition) and condition (place) should contribute to the processing of cases. Therefore, every transition (place) should be located on a path from place *i* to place *o.* The latter requirement corresponds to strongly connectedness if *o* is connected to *i* via an additional transition *t**. The Petri net shown in Figure 1 is a workflow net: *start* is the input place and *ready* is the output place. Note that the definition of a workflow net is a syntactical definition, the requirements can verified statically because they only relate to the structure of the Petri net.

The *soundness property* relates to the dynamics of the workflow process definition. A workflow net is sound if the following requirements are satisfied:

- For any case, it is possible to terminate, i.e., it is possible to reach a state with at least one token in the output place *o*.
- The moment the case terminates (i.e. a token appears in *o*), there are no tokens left behind in the workflow net. This means that there will be no dangling references.
- There are no dead tasks, i.e., starting with a token in the input place *i*, it should be possible to execute an arbitrary task by following the appropriate route through the WF-net.

Soundness is the minimal property any workflow process definition should satisfy. Note that soundness implies the absence of livelocks and deadlocks. Consider for example Figure 1. Clearly, the workflow net is not sound. Figure 4 shows a modified version of the workflow process definition shown in Figure 1. This modified workflow net is sound.



• Figure 4: A sound workflow process definition.

For a given workflow net, Woflan is able to decide whether it is sound. For this purpose Woflan uses an interesting relation between soundness on the one hand and liveness and boundedness on the other hand. A workflow net is sound, if and only if, the net obtained by connecting *o* and *i* via an additional transition *t** is live and bounded (see [5]). Although soundness can be decided in polynomial time for certain subclasses (e.g. by using the Rank theorem for free choice nets), Woflan constructs the reachability graph to verify whether the workflow net is live and bounded. For normal workflow process definitions, the size of the reachability graph is not a restricting issue. Woflan can cope with workflow process definitions with more than 200.000 states.

If a workflow process definition is sound, there is often no real reason to analyze it in more detail. Nevertheless, Woflan warns for constructs which look suspicious. Consider for example the workflow net shown in Figure 4. Woflan will give two warnings:

- The non-free-choice construct which involves place *c5* is reported. Woflan warns for non-free-choice constructs because they cannot be handled by many workflow management systems and they often correspond to a mixture of choice and synchronization.
- Woflan also warns for the fact that the AND-split *register* is complemented by the OR-join *c5*, i.e., there are two disjoint paths leading from the transition *register* to the place *c5*. Such a construct may result in two parallel flows which are not synchronized properly.

Despite these warnings the workflow net shown in Figure 4 is sound. However, Woflan warns for the use of the advanced construct involving *process_complaint* and *c5*.

If the workflow process definition is not sound, Woflan guides the user in finding and correcting the error. Consider for example the definition shown in Figure 1. For this workflow net, Woflan gives the following diagnostics:

- Woflan points out the fact that place *c8* is not bounded in the net extended with the transition *t** which connects the output place *ready* with the input place *start*. This means that it is possible to terminate and leave a token in *c8* (i.e. a dangling reference).
- The OR-split *c3* is complemented by the AND-join *archive*, i.e., there are two disjoint paths (one via *c8*) leading from the place *c3* to the transition *archive*. Such a construct may lead to a potential deadlock. In this case it does!
- Woflan reports that the workflow net is not covered by state machines (S-components), i.e., the net is not S-coverable. In fact, Woflan indicates that *c8* is the only place not in any S-component.
- The fact that something is wrong with c8 is also highlighted by the fact that place c8 is not in the support of any of the semi-positive place invariants generated by Woflan.

The above diagnostics clearly show that the optional synchronization of the two parallel flows via place *c8* is the source of the error. Removing *c8* or replacing *c8* by the construct shown in Figure 4 solves this problem and results in a sound workflow process definition. For a small workflow with only 8 tasks these results may seem trivial. However, workflows encountered in practice may have up to a 100 tasks. Experience shows that for workflows with more than 20 tasks it is not easy to locate the source of the error if the workflow net is not sound. Therefore, the support offered by Woflan is of the utmost importance for the verification of workflow process definitions.

To assist the user in repairing the error, Woflan offers an on-line help facility. The on-line help is based on a step-wise approach to locate and remove constructs which violate the soundness property. This enables users without a background in Petri nets to operate the tool and repair an erroneous workflow process definition.

For more information on the analysis techniques supported by Woflan, the reader is referred to [4] and [5]. For more information on the technical aspects of the tool Woflan, see [7] and [14].

5. Conclusion

In this paper we presented a tool based on Petri nets: Woflan. Woflan is an analysis tool which can be used to verify the correctness of a workflow procedure. The analysis tool uses state-of-the-art techniques to find potential errors in the definition of a workflow procedure and gives workflow designers a handle to construct correct workflows. Woflan is both from a theoretical and a practical point of view an interesting tool. On the one hand, Woflan uses advanced analysis techniques. On the other hand, it interfaces with some of leading workflow tools on the Dutch market (COSA/Protos). Woflan clearly shows that the workflow market is a challenging application domain for Petri-net-based technology.

Acknowledgements

The author would like to thank all the people involved in the development of Woflan, in particular Eric Verbeek and Dirk Hauschildt.

References

1. W.M.P. van der Aalst. Petri-net-based Workflow Management Software. In A. Sheth, editor, Proceedings of the NFS Workshop on Workflow and Process Automation in Information Systems, pages 114-118, Athens, Georgia, May 1996.
2. W.M.P. van der Aalst. Structural Characterizations of Sound Workflow Nets. Computing Science Reports 96/23, Eindhoven University of Technology, Eindhoven, 1996.
3. W.M.P. van der Aalst. Three Good reasons for Using a Petri-net-based Workflow Management System. In S. Navathe and T. Wakayama, editors, Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC'96), pages 179-201, Camebridge, Massachusetts, Nov 1996.
4. W.M.P. van der Aalst. Exploring the Process Dimension of Workflow Management. Computing Science Reports 97/13, Eindhoven University of Technology, Eindhoven, 1997.
5. W.M.P. van der Aalst. Verification of Workflow Nets. In P. Azema and G. Balbo, editors, Application and Theory of Petri Nets 1997, volume 1248 of Lecture Notes in Computer Science, pages 407-426. Springer-Verlag, Berlin, 1997.
6. W.M.P. van der Aalst and K.M. van Hee. Workflow Management: Modellen, Methoden en Systemen (in Dutch). Academic Service, Schoonhoven, 1997.

7. W.M.P. van der Aalst, H.M.W. Verbeek and D. Hauschildt. A Petri-net-based Tool to Analyze Workflows. In B. Farwer, D.Moldt, and M.O. Stehr, Petri Nets in System Engineering (PNSE'97), pages 78-89, FBI-HH-B-205/97, University of Hamburg, Sept. 1997.
8. Bakkenist Management Consultants. ExSpect 5.0 User Manual, 1996.
9. K. Barkaoui, J.M. Couvreur, and C. Dutheillet. On liveness in Extended Non Self-Controlling Nets. In G. De Michelis and M. Diaz, editors, Application and Theory of Petri Nets 1995, volume 935 of Lecture Notes in Computer Science, pages 25-44. Springer-Verlag, Berlin, 1995.
10. J. Desel and J. Esparza. Free choice Petri nets, volume 40 of Cambridge tracts in theoretical computer science. Cambridge University Press, Cambridge, 1995.
11. C. Ellis, K. Keddara, and G. Rozenberg. Dynamic change within workflow systems. In N. Comstock and C. Ellis, editors, Conf. on Organizational Computing Systems, pages 10 - 21. ACM SIGOIS, ACM, Aug 1995. Milpitas, CA.
12. C.A. Ellis and G.J. Nutt. 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-Verlag, Berlin, 1993.
13. J. Esparza and M. Silva. Circuits, Handles, Bridges and Nets. In G. Rozenberg, editor, Advances in Petri Nets 1990, volume 483 of Lecture Notes in Computer Science, pages 210-242. Springer-Verlag, Berlin, 1990.
14. D. Hauschildt, H.M.W. Verbeek, and W.M.P. van der Aalst. WOFLAN: a Petri-net-based Workflow Analyzer. Computing Science Reports 97/12, Eindhoven University of Technology, Eindhoven, 1997.
15. P. Lawrence, editor. Workflow Handbook 1997, Workflow Management Coalition. John Wiley and Sons, New York, 1997.
16. G. De Michelis, C. Ellis, and G. Memmi, editors. Proceedings of the second Workshop on Computer-Supported Cooperative Work, Petri nets and related formalisms, Zaragoza, Spain, June 1994.
17. T. Murata. Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE, 77(4):541-580, April 1989.
18. Pallas Athena. Protos User Manual. Pallas Athena BV, Plasmolen, The Netherlands, 1997.
19. Software-Ley. COSA User Manual. Software-Ley GmbH, Pullheim, Germany, 1996.
20. WFMC. Workflow Management Coalition Terminology and Glossary (WFMC-TC-1011). Technical report, Workflow Management Coalition, Brussels, 1996.
21. M. Wolf and U. Reimer, editors. Proceedings of the International Conference on Practical Aspects of Knowledge Management (PAKM'96), Workshop on Adaptive Workflow, Basel, Switzerland, Oct. 1996.