Appears in: Proceedings of the 7th European Concurrent Engineering Conference (ECEC'2000)
Leicester, United Kingdom, April 17-19 2000

The Workflow Management Coalition Standard WPDL: First Steps towards Formalization

Stefan Junginger
University of Vienna
Dep. Knowledge Engineering
Brünnerstr. 72
A-1210 Vienna, Austria
E-mail: sjung@dke.univie.ac.at

KEYWORDS

Workflow Management Systems, Workflow Management Coalition, WPDL, Graph Theory

ABSTRACT

WPDL (Workflow Process Definition Language) is a file format for exchanging process definitions between workflow management systems (WMS) and process modeling tools. It is a standard defined by the WfMC (Workflow Management Coalition) and was published in November 1998. Unfortunately, WPDL in its current status (V.1.1) lacks a formal definition. This is seen as one of the main reasons why WPDL is not yet widely supported by modeling tools and WMS. This paper makes a proposal how WPDL can be formalized using graph theory. It is explained how this formal model can be used in defining the flow semantics of WPDL and transformation algorithms between the different WPDL conformance classes.

INTRODUCTION

To achieve a high degree of concurrency in product development, an effective communication and coordination support is needed (Herbst 1999). Workflow management systems (WMS) enable the execution of repetitive processes by enacting them according to a predefined process model, e.g. of the product development process. In order to decrease the risk of using a concrete workflow product, standards are a necessity. Therefore, in 1993 the standardization body Workflow Management Coalition (WfMC) was founded (Lawrence 1997; WfMC 1999a). Currently, the WfMC has more than 225 member organizations covering all roles within workflow technology (vendors, users, consultants, and research institutions). In 1995 the WfMC reference model was published, defining five classes of interfaces for WMS (Hollingsworth 1995). Interface 1 is related to "Process Definition Import and Export". Until now modeling tools and WMS use their own proprietary languages for process modeling which makes the exchange of models extremely difficult. The first official version of *WPDL (Workflow Process Definition Language)* was published in November 1998, the current version is V. 1.1 (WfMC 1999c). WPDL is based on a process definition meta model which describes the core objects within a process definition, their relations and attributes (see figure 1). It is designed as a file based exchange format and specified by an EBNF grammar.

According to (Curtis et al. 1992), processes can be characterized by the following views (see also (Jablonski 1994)):

1. *Functional view:* Which activities/processes are performed?
2. *Behavioral view:* When are activities/processes performed ("control of the process")?
3. *Organizational view:* Who performs an activity/process?
4. *Informational view:* Which information elements are produced or consumed by activities/processes?

In the following, we concentrate mainly on the functional and the behavioral view, i.e. the entities "Workflow Process Activity" and "Transition Information" (see figure 1).

Currently, WPDL is not yet widely supported by modeling tools and WMS (WfMC 1999b). The main reasons for this are probably:

1. WPDL lacks a formal definition as is common for the specification of programming languages. This makes the implementation and usage of WPDL problematic. A formalization of WPDL would also enable the definition of its flow semantics.
2. Current WMS greatly differ concerning their functionality and therefore also concerning their modeling concepts. For this purpose extensibility mechanisms and conformance classes in WPDL have been defined (WfMC 1999c, p. 21, p. 44). The conformance classes in particular have not been investigated in detail ("Is the exchange of WPDL between different conformance classes possible? If yes, how?").

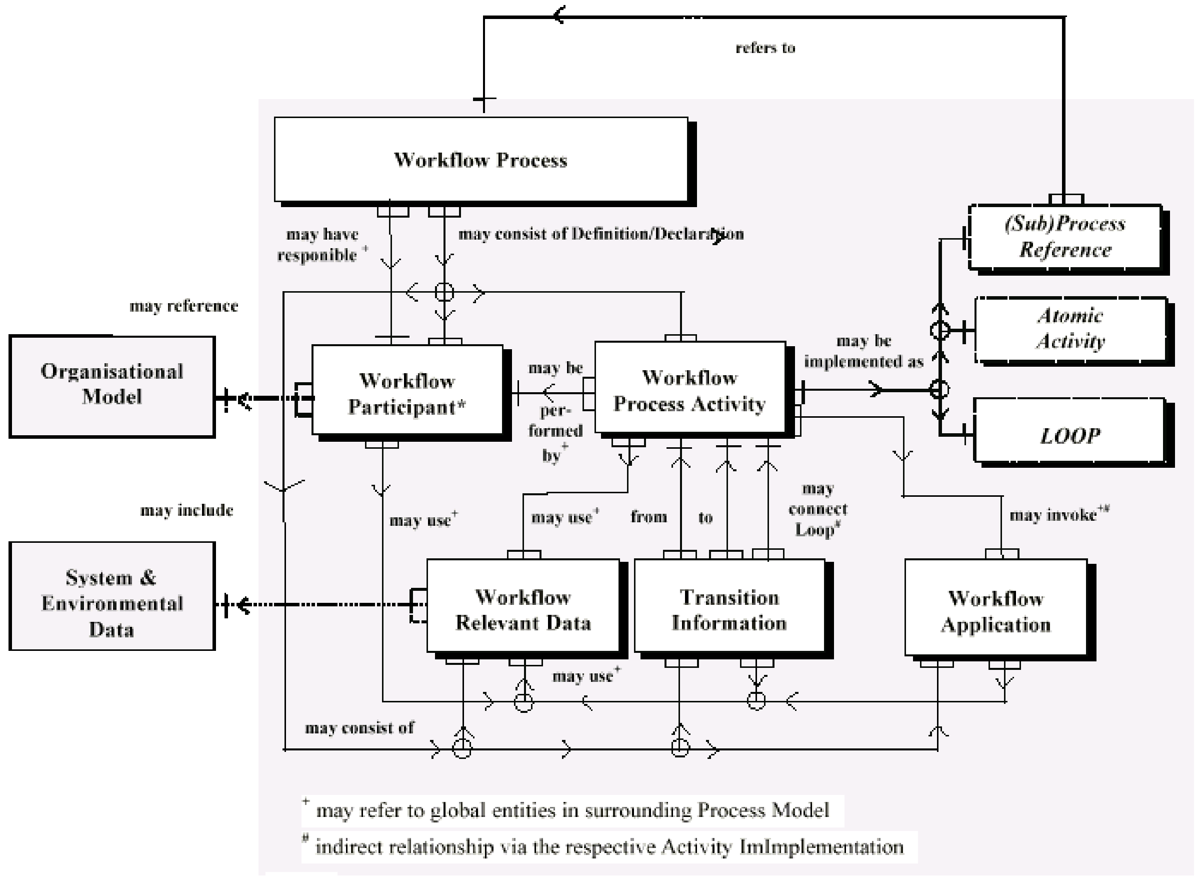

Figure 1: WfMC workflow process definition meta model (WfMC 1999c, p. 27)

3. WPDL is not yet widely known amongst users, therefore there is not enough pressure on the vendors to support WPDL.

This paper mainly deals with item 1. It makes a proposal on how WPDL can be formalized. Additionally, some interrelations between the WPDL conformance classes are shown (item 2). These interrelations are of great interest for vendors as well as for WMS users. The better WMS users understand WPDL the more they will be able to force vendors to support WPDL (item 3).

The rest of the paper is structured as follows: The next section describes the main ideas of a formalization of WPDL based on graph theory. Then, approaches for defining the flow semantics of WPDL are presented. Afterwards, algorithms for the transformation between the different WPDL conformance classes are discussed. The paper concludes with a summary and an outlook to future work.

APPLYING GRAPH THEORY TO WPDL

As most process modeling tools and WMS use directed graphs (*digraphs*) for modeling processes it seems sensible to use graph theory as the basis for a formalization (Junginger 1998; Karagiannis et al. 1996; Leymann and Altenhuber 1994; Rump 1997). The behavioral view on a WPDL process definition can be interpreted as a digraph $G = (A, T)$, where A is the set of *nodes* (*Workflow Process Activities*) and $T \subseteq A \times A$ is the set of *edges* (*Transition Information*). The edges are labeled with a specific attribute *Condition* which contains a predicate that is evaluated during execution. In WPDL control information is specified using certain attributes of activities (WfMC 1999c, p. 43). There are distinguished *AND* and *XOR split* and *join* elements (see figure 2).

Here split and join elements are seen as extra nodes as this makes the specification of syntactic restrictions and algorithms easier.

Similarly to split and join elements, so-called *blocks* can be defined using specific activity attributes (see figure 3). A block represents a "bracket" for parts of a process definition which are connected (WfMC 1999c, p. 41). The (optional) begin and end block elements of an activity are also seen as extra nodes here.

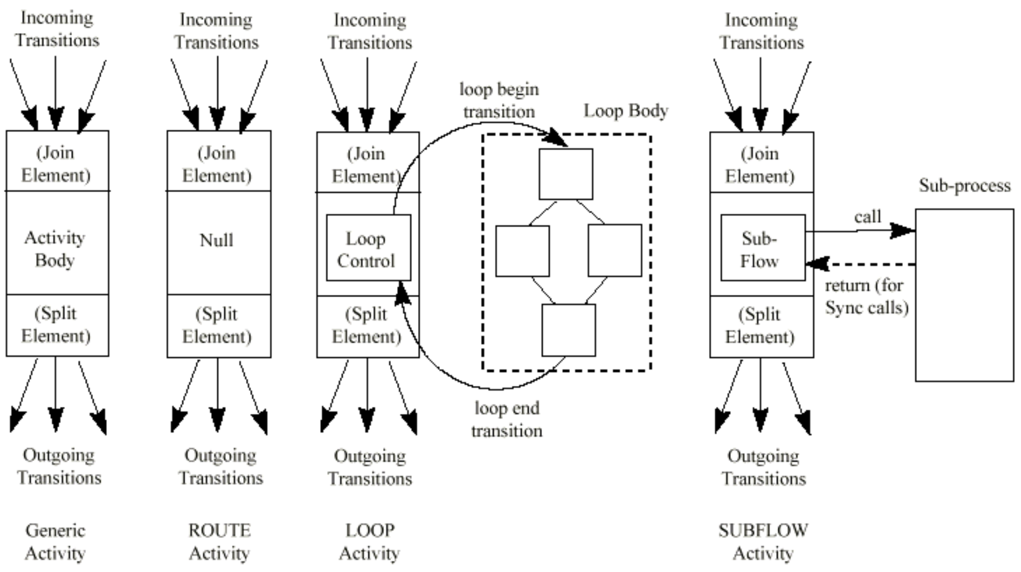

Figure 2: Activity structures and transition conditions (WfMC 1999c, p. 33)

Additionally, WPDL defines so-called *Route Activities* which are "dummy" activities that can be used purely for routing purposes, i.e. they contain neither a performer nor an application and their execution has no effect on workflow relevant data or application data (WfMC 1999c, p. 35).

In the workflow process definition meta model, three subclasses of the class "Workflow Process Activity" are distinguished (see figure 1 and figure 2):

1. *(Sub)Process References* are activities which are refined as a subprocess. They contain a reference to another Workfow Process Definition which is enacted (synchronously or asynchronously) when the (Sub)Process Reference is reached during execution (WfMC 1999c, p. 39).
2. *Atomic Activities* are atomic from the workflow engines' point of view. They have a performer (*Workflow Participant*) and a *Workflow Application* is assigned (WfMC 1999c, p. 37).
3. *Loop Activities* are activities that are refined as a loop. The loop body (as a set of activities) is connected to the loop activity via so-called loop connecting transitions. WHILE and REPEAT_UNTIL loop activities are distinguished (WfMC 1999c, p. 39).

Using these activity types the definition given above can be refined. The behavioral view on a WPDL process definition is now defined as:

$$G = (A, S_{AND}, S_{XOR}, J_{AND}, J_{XOR}, B_{begin}, B_{end}, T_{flow}, T_{begin\_loop}, T_{end\_loop}, T_{begin\_block}, T_{end\_block})$$

where:

- $A = A_S \cup A_A \cup A_L \cup A_R$ is the set of activities, where:
  - A_S is the set of (sub)process references,
  - A_A is the set of atomic activities,
  - A_L is the set of loop activities,
  - A_R is the set of route activities,
- S_{AND} is the set of AND splits,
- S_{XOR} is the set of XOR splits,
- J_{AND} is the set of AND joins,
- J_{XOR} is the set of XOR joins,
- B_{begin} is the set of begin block nodes,
- B_{end} is the set of end block nodes,
- $T_{flow} \subseteq V \times V$ $(V = A \cup S_{AND} \cup S_{XOR} \cup J_{AND} \cup J_{XOR} \cup B_{begin} \cup B_{end})$ is the set of "normal" transitions between the nodes.
- $T_{begin\_loop} \subseteq A_L \times V$ is the set of transitions from loop activities to the first node(s) of the corresponding loop bodies.
- $T_{end\_loop} \subseteq V \times A_L$ is the set of transitions from the last node(s) of the loop bodies to the loop corresponding activities.
- $T_{begin\_block} \subseteq B_{begin} \times B_{end}$ is the set of connections from begin block nodes to the corresponding end block nodes.
- $T_{end\_block} \subseteq B_{end} \times B_{begin}$ is the set of connections from end block nodes to the corresponding begin block nodes.

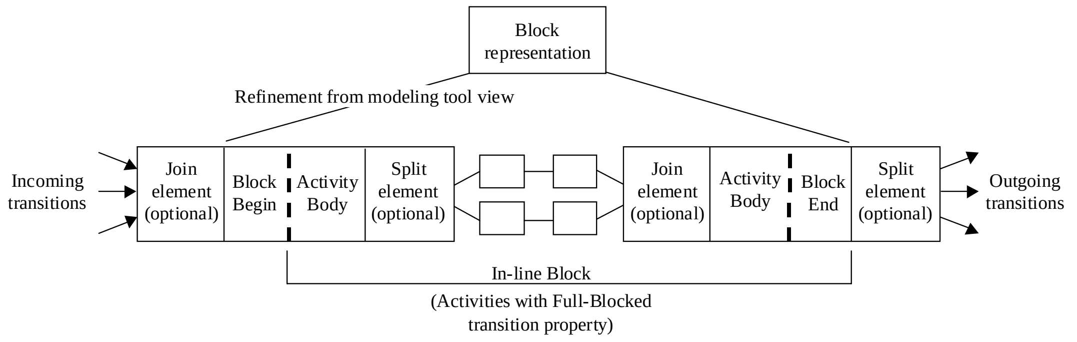

Figure 3: Block structure (WfMC 1999c, p. 42)

Using graph theory concepts, restrictions on G can be defined, for example $|\{a \in V \mid (a,x) \in T_{flow}\}| \leq 1$ for all $x \in A \cup J_{AND} \cup J_{XOR} \cup B_{begin} \cup B_{end}$ (only AND splits and XOR splits can have more than one successor). Depending on the definition of the flow semantics - which is explained in the next section - additional restrictions have to be defined.

DEFINING THE FLOW SEMANTICS

The formalization described in the last section can be used as a basis for a definition of the flow semantics of WPDL. For this, the minimal state model of process instances described in the WPDL specification can be used (WfMC 1999c, p. 11). Figure 4 and 5 show possible state machines for process and node instances.

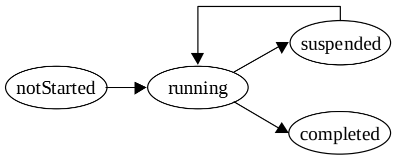

Figure 4: Possible state machine for process instances

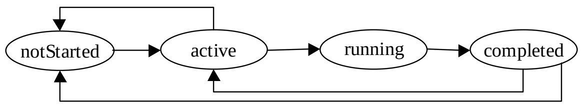

Figure 5: Possible state machine for node instances

Similarly to the "Token Game" in Petri Net theory, the process execution can be visualized by tokens which are floating through the process (Reisig 1986). This is depicted in figure 6. The tokens represent the different states a node can have.

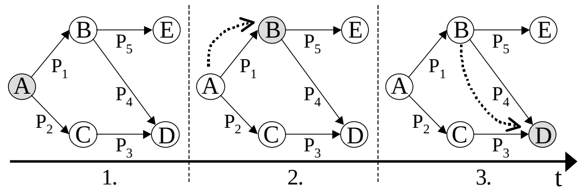

Figure 6: Process execution

Different approaches can be chosen to define the flow semantics of WPDL (and any other process modeling language):

- *Operational definition:* Specification of the algorithm of an execution engine, e.g. in pseudo code.
- *Comparative definition:* Mapping of the elements of WPDL to a process modeling language of which the flow semantics are defined, e.g. Petri Nets. The chosen process modeling language has to be at least as powerful as WPDL. An example can be found in (Langner et al. 1997).
- *Descriptive definition:* Specification of rules that describe under which conditions the process and the node instances take which state. This requires again an (abstract) engine of which the execution algorithm is known. Such an abstract execution engine is for example described in (Junginger 1998).

Each of these approaches has different advantages and disadvantages. Independently of which approach is chosen, the handling of AND splits and joins should be considered. Figure 7 shows two cases where it is extremely difficult to define the flow semantics. It is not clear under which conditions the AND join should be activated. The same problem for EPC (Event Process Chains) is described in (Rittgen 1999). One solution is to use the same restrictions as in the business process management tool ADONIS[1] (BOC 1999): AND splits and joins should form a block, this means that no transitions are allowed to "leave" this block.

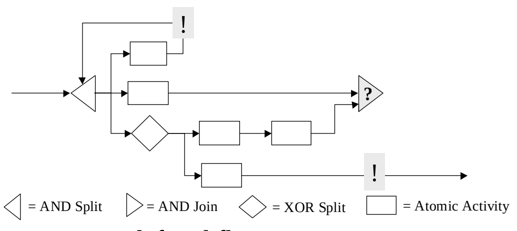

Figure 7: Undefined flow semantics at AND joins

[1] ADONIS is a registered trademark of BOC GmbH.

ALGORITHMS FOR THE WPDL CONFORMANCE CLASSES

The WPDL specification distinguishes three conformance classes (WfMC 1999c, p. 44):

- *NON-BLOCKED:* There is no restriction for this class.
- *LOOP-BLOCKED:* For cycles only loop activities are used, i.e. the graph is acyclic.
- *FULL-BLOCKED:* For each (AND or XOR) join there is a corresponding (AND or XOR) split and these tuples are also defining blocks. Additionally, in an AND split no conditions are permitted and in XOR splits either an unconditional or OTHERWISE transition is required.

Obviously, LOOP-BLOCKED WPDL is a subset of NON-BLOCKED WPDL. If the graph is acyclic, i.e. only loop activities are used for specifying cycles, FULL-BLOCKED WPDL is a subset of LOOP-BLOCKED WPDL (see figure 8). This is probably intended in the WPDL specification but not explicitly stated.

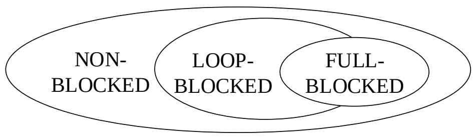

Figure 8: WPDL conformance classes and their interrelations

This raises the following questions:

1. Can NON-BLOCKED WPDL be converted to LOOP-BLOCKED WPDL? If yes, which preconditions have to be fulfilled?
2. Can LOOP-BLOCKED WPDL be converted to FULL-BLOCKED WPDL (which is still LOOP-BLOCKED)? If yes, which preconditions have to be fulfilled?

Using *strongly connected components* of digraphs helps in answering question 1.[2] A strongly connected component of a graph $G=(A,T)$ is a subgraph $G'=(A',T')$, $A'\subseteq A$ and $T'\subseteq T$, where each node of A' can be reached from any other node of A' by following the edges. An algorithm for computing the strongly connected components of a given digraph can be found for example in (Ottmann and Widmayer 1990, pp. 566-570). If a strongly connected component *SCS* in a WPDL process definition contains exactly one "incoming node", this means exactly one node which has incoming edges $t\in T$ from nodes of the set $A-A'$, and exactly one "outgoing node", which means exactly one node which has outgoing edges $t\in T$ to nodes of the set $A-A'$, *SCS* can be converted to a loop activity where *SCS* forms the body of the loop activity (see figure 9).

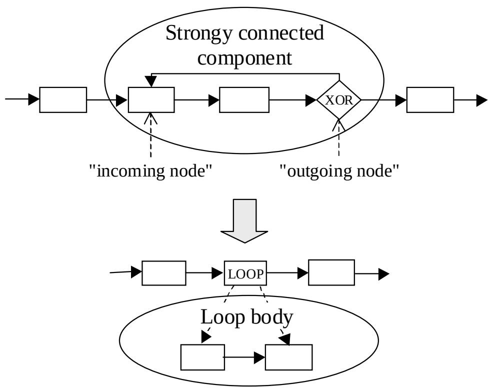

Figure 9: Transformation of strongly connected components to loop activities

Similarly to the transformation of strongly connected components into loop activities, other graph algorithms can be used for transforming WPDL into FULL-BLOCKED WPDL, i.e. to identify places in the graph where block definitions have to be inserted. Due to space limitations this is omitted here.

SUMMARY AND FUTURE WORK

This paper made a proposal of how to formalize WPDL based on graph theory and explained how this formal model can be used in defining the flow semantics and transformation algorithms between the different WPDL conformance classes. The paper focused on the functional and behavioral views on WPDL, the other views (respectively the entities "Workflow Participant", "Workflow Relevant Data", and "Workflow Application") are subject to further work.

Route activities and the WPDL conformance classes make it easy to generate WPDL out of most process modeling languages. For importing for example NON-BLOCKED WPDL when the target process modeling language is based on LOOP-BLOCKED concepts, the presented algorithm is needed. Therefore it is proposed that such types of algorithms should be made part of the standard. This would ease the implementation of WPDL. When these algorithms are well known, a future option could be to remove the conformance classes from the standard.

Some missing concepts of WPDL are (see also (Sheth and Miller 1998)):

1. Time modeling
2. Events

[2] In (WfMC 1998) the semantics of loop activities are explained by replacing them by WPDL parts without loop activities. This explanation is not part of the WPDL specification. The algorithm presented here can be seen as an inversion.

3. Organizational modeling
4. Process history cannot be accessed in the workflow participants definition

The items 3 and 4 are addressed by a WfMC working group that was founded at the end of 1998 and deals with organization/resource modeling. Additionally, WPDL might profit from other efforts for defining process exchange formats such as PSL and PIF (Lee et al. 1997; Tissot and Gruninger 1999).

Obviously, WPDL is not yet "perfect". It is a first step towards the reusability of process definitions that should be supported in particular by the users. However, as a basis for extensions of WPDL a formalization is needed. Otherwise every WPDL extension will become more difficult to understand and to implement.

REFERENCES

BOC GmbH. 1999. *ADONIS Version 3.0 – Users Guide*. Vienna.

Curtis, B.; M. I. Kellner; and J. Over. 1992. "Process Modeling." *Communications of the ACM* 35, no. 9: 75-90.

Herbst, J. 1999. "Inducing Workflow Models from Workflow Instances." In *Proceedings of the 6th European Concurrent Engineering Conference*. Society for Computer Simulation (SCS), 175-182.

Hollingsworth, D. 1995. "Workflow Management Coalition. The Workflow Reference Model. WFMC-TC-1003, 19-Jan-95, 1.1." http://www.aiim.org/wfmc/standards/docs/tc003v11.pdf (1999-12-06).

Jablonski, S. 1994. "Functional and Behavioral Aspects of Process Modelling in Workflow Management Systems." In *Workflow Management. Challenges, Paradigms and Products*, G. Chroust and A. Benczúr, eds., Oldenbourg, 113-133.

Junginger, S. 1998. "Eine operationale Ablaufsemantik für graphenbasierte Prozeßmodellierungsmethoden." BPMS-Bericht. University of Vienna, July 1998. (in German).

Karagiannis, D.; S. Junginger; and R. Strobl. 1996. "Introduction to Business Process Management Systems Concepts." In *Business Process Modelling*, B. Scholz-Reiter and E. Stickel, eds., Springer, 81-106.

Langner, P.; C. Schneider; and J. Wehler. 1997. "Prozeßmodellierung mit ereignisgesteuerten Prozeßketten (EPKs) und Petri-Netzen." *Wirtschaftsinformatik* 39, no. 5: 479-489. (in German).

Lawrence, P. 1997. *Workflow Handbook 1997*. John Wiley & Sons.

Lee, J.; M. Gruninger; Y. Jin; T. Malone; A. Tate; G. Yost et al. 1997. "PIF: The Process Interchange Format v.1.2. December 1997." http://spot.colorado.edu/~jintae/pif/pif12.rtf (1999-12-07).

Leymann, F. and W. Altenhuber. 1994. "Managing Business Processes as an Information Resource." *IBM Systems Journal* 33, no. 2: 326-348.

Ottmann, T. and P. Widmayer. 1990. *Algorithmen und Datenstrukturen*. BI Wissenschaftsverlag. (in German).

Reisig, W. 1986. *Petrinetze. Eine Einführung*. 2nd Edition, Springer. (in German).

Rittgen, P. 1999. "Quo vadis EPK in ARIS? Ansätze zu syntaktischen Erweiterungen und einer formalen Semantik." Appears in *Wirtschaftsinformatik*. Accesible in the members area of the Wirtschaftsinformatik hompage: http://www.wirtschaftsinformatik.de (1999-12-06). (in German).

Rump, F. 1997. "Erreichbarkeitsgraphbasierte Analyse ereignisgesteuerter Prozeßketten." Technical Report, University of Oldenburg, April 1997, http://www-is.informatik.uni-oldenburg.de/~rump/paper/analyse/analyse.ps (1999-06-16). (in German).

Sheth, A. and J. Miller. 1998. "Workflow Management: Technology, Experiences and Research." Tutorial, 14th Intl. Conf. on Data Engineering, February 23, 1998, Orlando FL, http://orion.cs.uga.edu:5080/workflow/presentation/DE-tutorial-98/ (1999-11-01).

Tissot, F. and M. Gruninger. 1999. "PSL Informal Documentation. Version 0.1." http://www.mel.nist.gov/psl/p4/990330informal.rtf (1999-12-07).

WfMC. 1998. "Loop Semantics." WfMC Internal Paper. 4th October 1998. Accessible in the members area of the WfMC homepage (http://www.wfmc.org).

WfMC. 1999a. "Introduction to the Workflow Management Coalition." http://www.wfmc.org (1999-12-06).

WfMC. 1999b. "WfMC - Conformance to Interface Standards." http://www.aiim.org/wfmc/standards/conformance.htm (1999-12-06).

WfMC. 1999c. "Workflow Management Coalition: Workflow Management Coalition. Interface 1: Process Definition Interchange. Process Model. Document Number WfMC TC-1016-P, Document Status - Version 1.1 (Official release), Issued on October 29, 1999." http://www.aiim.org/wfmc/standards/docs/if19910v11.pdf (2000-02-21).