A Survey of Comparative Business Process Modeling Approaches

Ruopeng Lu, Shazia Sadiq

School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane, QLD, 4072, Australia
{ruopeng, shazia}@itee.uq.edu.au

Abstract. There has been a huge influx of business process modeling languages as business process management (BPM) and process-aware information systems continue to expand into various business domains. The origins of process modeling languages are quite diverse, although two dominant approaches can be observed; one based on graphical models, and the other based on rule specifications. However, at this time, there is no report in literature that specifically targets a comparative analysis of these two approaches, on aspects such as the relative areas of application, power of expression, and limitations. In this paper we have attempted to address this question. We will present both a survey of the two approaches as well as a critical and comparative analysis.

Key words: business process management, workflows, business process modeling and analysis, rule-based workflows, graph-based workflows.

1 Introduction

Business process management (BPM) solutions have been prevalent in both industry products and academic prototypes since the late 1990s. It has been long established that automation of specific functions of enterprises will not provide the productivity gains for businesses unless support is provided for overall business process control and monitoring. Business process modeling is the first and most important step in BPM lifecycle [5], which intends to separate process logic from application logic, such that the underlying business process can be automated [32]. Typically, process logic is implemented and managed through a business process management system (BPMS) and application logic through underlying application components.

Business process modeling is a complicated process and it is obvious that different modeling approaches have their strengths and weaknesses in different aspects due to the variety of their underlying formalisms. There are many well-known problems regarding process modeling methodologies, such as the classic tradeoff between expressibility of the modeling language and complexity of model checking. Some languages offer richer syntax sufficient to express most relevant business activities and their relationships in the process model, while some provide more generic modeling constructs which facilitate efficient verification of the process model at design time. These have been prevalent in research prototypes (e.g., FlowMake [30], ADEPT$_{flex}$ [29], YAWL [3]), in commercial products (e.g., Tibco Staffware Process Suite [34], Oracle BPEL Process Manager [28], ILOG BPM [15]), as well as in industrial standard modeling languages (BPEL4WS [26], BPMN [27]).

Among the huge options of modeling languages, there have been methodical investigations in literature that attempt to address a variety of issues. These investigations involve a number of comparison techniques. First, in [33], an **empirical study** on process modeling success in industry is presented, where success factors of process modeling are generalized from multiple case studies of industry applications and the measure for effective process modeling is derived. Second, **ontological comparison** techniques utilize the semantic richness of an appropriate ontology as the benchmark for comparing process modeling languages. In [12], the interoperability of a business process specification (in particular, ebXML) is studied through a mapping from constructs in Bunge-Wand-Weber (BWW) ontology model to constructs in ebXML. Lastly, [20] presents a framework for selecting appropriate process modeling tools based on the **heuristics** collected from process modeling and business domain experts. The heuristics is used to provide quantifiable measure for indicating preferences on modeling tools selection.

We have conducted a study on the comparative business process modeling languages [23] based on a different comparison criteria. The scope of the comparison is on the most critical dimension of business process models, namely **control flow** perspective [30], from a selection of modeling approaches based on different theoretical foundations. The two most dominant foundations can be found in models bases on graphs and rules. The goal of comparison is to investigate, through the language representatives, the strengths and limitations of different theoretical foundations when being applied in business process modeling.

The focus of this paper is to summarize the comparison results and critical remarks reported in [23], and to facilitate future investigations and developments on business process modeling. In what follows, a survey of business process modeling approaches is first presented in section 2 to provide insights into current process modeling practices, based on which the comparison methodology is discussed in section 3. The comparison results, along with critical remarks are presented in section 4. Process modeling techniques in current commercial BPMS products are also briefly discussed in section 5 to present industry developments and trends. We conclude this paper and discuss possible future work in section 6.

2 A Survey on Business Process Modeling Approaches

The objective of process modeling is to provide high-level specification independent from the implementation of such specification. In this paper, we use the following definition for process modeling languages: A **process modeling language** provides appropriate syntax and semantics to precisely specify business process requirements, in order to support automated process verification, validation, simulation and process automation. The syntax of the language provides grammar to specify objects and their dependencies of the business process, often represented as a language-specific

process model, while the semantics defines consistent interpretation for the process model to reflect the underlying process logic.

It is essential that a process model is properly defined, analyzed, and refined before being deployed in the execution environment. In a narrower scope, business process modeling can be referred to as workflow modeling, as workflow management systems (WFMS) provide equivalent functionalities to BPMS in business process modeling, analysis and enactment.

It has been found that there are two most predominant formalisms on which process modeling languages are developed, namely **graph-based** formalism and **rule-based** formalism. A graph-based modeling language has its root in graph theory or its variants, while a rule-based modeling language is based on formal logic.

2.1 Graph-Based Process Modeling Approaches

In a graph-based modeling language, process definition is specified in graphical process models, where activities are represented as nodes, and control flow and data dependencies between activities as arcs. The graphical process models provide explicit specification for process requirements.

Most graph-based languages have their root in Petri Net theory, which was applied in workflow modeling for the first time in 1977 by Zisman [40]. Many process modeling languages have been proposed based on different variants of Petri Nets to provide extra expressibility and functionality since then, including High Level Petri Nets [11], Low Level Petri Nets [36], and Colored Petri Nets [24]. More details can be found in a survey on Petri Net applications in workflow modeling by Jenssens et al [16].

The strengths of Petri Net based modeling approaches include formal semantics despite the graphical nature, and abundance of analysis techniques [1]. Formal methods [2] have been provided for specifying, analyzing and verifying the properties of static workflow structures (e.g. state transitions, deadlocks).

On the other hand, there are many graph-based modeling languages that carry similar advantages of Petri Net based languages, which also have simple and easy-to-understand syntax and semantics.

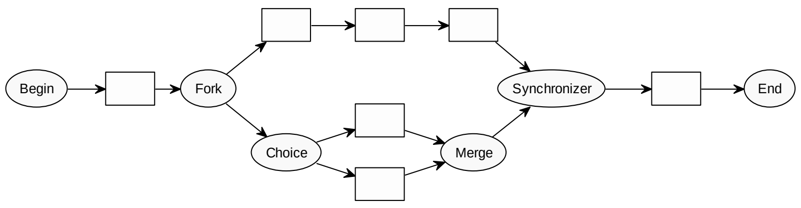

Fig. 1. A graph-based process model in FlowMake [30] syntax.

For example, the syntax of FlowMake language [30] (cf. Fig. 1) contains 3 types of objects, task, coordinator and transition. A **task** represents a unit of work to be done (denoted by a rectangle). A **coordinator** is used to define how tasks are scheduled (denoted by an oval), which is further divided into begin, end, choices, merge, fork and synchronizer. A **transition** links any two nodes (task or coordinator) and is represented by a directed arc.

Table 1 provides a list of representative graph-based modeling approaches.

Table 1. Graph-based modeling approaches.

Authors	Approach	Brief Description
Sadiq & Orlowska [30]	FlowMake	FlowMake is a design and analysis methodology for workflow modeling, which includes a set of constraints to verify the syntactic correctness of the graphical workflow specifications by a graph-reduction algorithm.
Reichert & Dadam [29]	$\text{ADEPT}_{\text{flex}}$	$\text{ADEPT}_{\text{flex}}$ is a graph-based modeling methodology which supports ad hoc changes to process schema. A complete and minimal set of change rules is given to preserve the correctness and consistency property, which provides a comprehensive solution for applying complex and dynamic structural changes to a workflow instance during its execution.
Casati et al [8]	Conceptual Modeling	The conceptual modeling approach divides a business process into workflow tasks (WT) and workflow (WF). A workflow execution architecture is proposed, which supports syntax-directed translation from workflow definition to executable active-rules, and provides operational semantics and an implementation scheme for many components of WFMS.
van der Aalst et al [3]	YAWL	YAWL is a Petri Net based workflow language, which supports specification of the control flow and the data perspective of business processes. The language has formal semantics that encompasses workflow patterns [26] to guarantee language expressibility.
Casati et al [9]	WIDE	WIDE is designed to support next-generation workflow management functionality in a distributed environment. The architecture is based on a commercial database management system as the implementation platform, with extended transaction management and active rule support.
Liu & Pu [22]	ActivityFlow	ActivityFlow provides a uniform workflow specification interface and helps increase the flexibility of workflow changes by supporting reasoning about correctness and security of complex workflow activities independently from the underlying implementation mechanism.

2.2 Rule-Based Process Modeling Approaches

Rule-based formalisms have a wide area of applications in BPM domain, such as workflow coordination [18, 19] and exception handling [6]. Our consideration is limited to rule-based approaches where logical rules are used to represent structural, data and/or resource dependencies between task executions in business processes.

In a typical rule-based approach, process logic is abstracted into a set of rules, each of which is associated with one or more business activity, specifying properties of the activity such as the pre and post conditions of execution. The processing entity (process enactment mechanism) is a rule inference engine. At runtime, the engine examines data and control conditions and determines the best order for executing relevant business activities according to pre-defined rules. The typical enactment mechanism for general rule-based workflows is the rule inference engine, which is capable for evaluating current process events and triggering further actions (i.e., analogous to a workflow engine in common understanding [37]).

In [13], rule-based systems have been proposed as the first practical methodology to capture and refine human expertise, and to automate reasoning for problem solving. In 1990s, active database systems with the basic mechanism of Event-Condition-Action (**E-C-A**) rules have been applied to WFMS for coordinating task execution [14, 19]. The E-C-A paradigm has since been the foundation for many but not most rule-based process modeling languages. A basic E-C-A rule has the following syntax:

ON event **IF** condition **DO** action

An **event** specifies the triggering operation when a rule has to be evaluated, which indicates the transformation from one execution state to the other. An event can also be a simple change of task/process execution state (e.g., task “submit maintenance request” completes execution), or upon complex business process events (e.g. 2 days after product delivery, or on every 50 funding cases approved by the same project manager). The **condition** is the pre-condition to be checked before triggering any subsequent action, typically be the availability or of certain value of some process relevant data (e.g., requested funding $\geq$ 50,000). An **action** can be the execution of certain tasks, or triggering evaluation of other E-C-A rules. The result of executing an action can as well raise an event. A simple E-C-A rule can be extended with an additional **Else Action** to indicate the subsequent activities if the condition is not satisfied.

At the same time, software agent technology has been applied to model and enact workflows to provide flexibility and expressibility in business process automation. An agent is a piece of autonomous software that can perform certain actions to fulfill the design goal. An agency is a collection of autonomous artificial agents that communicate and work collaboratively to realize the process goal [17]. In the agent-based approach, the processing entity is an agency (i.e., a collection of software agents), and logical expressions are used to regulate the behaviors of autonomous agents. Rules are used to regulate actions of agents, or serve as the shared knowledge base among collaborating agents.

Table 2 lists some representative rule-based modeling approaches.

Table 2. Rule-based modeling approaches.

Authors	Approach	Brief Description
Knolmayer et al [21]	E-C-A Based Business Rules	The approach provides an E-C-A rule-based process model to serve as an integration layer between multiple process modeling languages, as well as functionality to support refinement of business rules.
Zeng et al [38]	PLM_{flow}	PLM_{flow} provides a set of business inference rules which is designed to dynamically generate and execute workflows. The process definition is specified in business rule templates, which include backward-chain rules and forward-chain rules. The process instance schema is determined by the rule engine using backward-chain and forward-chain inference at runtime.
Kappel et al. [18]	Object-Rule-Role approach	The proposed framework supports reusability and adaptability using E-C-A rules to allocate tasks and resources in workflows.
Jennings et al [17]	ADEPT	The ADEPT system is an infrastructure for designing and implementing multi-agent systems for workflows. Process logic is expressed in the service definition language (SDL), which specifies services that give the agents sufficient freedom to take alternative execution paths at run-time to complete the process goal.
Müller et al [25]	AgentWork	AgentWork is a WFMS prototype based on agent technology, where agents are used for monitoring exceptional events. Reactive and predictive adaptations to workflow exceptions are defined through temporal E-C-A rules and automated by agents. A rule model is proposed for managing temporal E-C-A rules for workflows.
Zeng et al [39]	AgFlow	AgFlow contains a workflow specification model and the agent-based workflow architecture. The process definition is specified by defining the set of tasks and the workflow process tuple, where the control flow aspects can be reflected in the task specific E-C-A rules.

3 Comparison Methodology

The following methodology has been developed to conduct the comparative analysis presented in this paper [23]. First, a selection of process modeling approaches have been identified in literature, and classified according to two most prominent formalisms (graph-based and rule-based) in process modeling, through which a variety of control flow functionalities are displayed (cf. Section 2).

Second, a minimal set of comparison criteria is identified for charactering the functionality of **control flow** capabilities, considering the design time (process modeling) and runtime (execution) requirements. The most important criteria are briefly discussed as follows:

– **Expressibility**: the expressive power of a process modeling language that is governed by its ability to express specific process requirements reflecting the purpose of process modeling and execution. A process model is required to be complete, which should contain structure, data, execution, temporal, and transactional information of the business process [30, 32].
– **Flexibility**: the ability of the business process to execute on the basis of a loosely, or partially specified model, where the full specification is made at runtime [31].
– **Adaptability**: which is the ability of the workflow processes to react to exceptional circumstances, which may or may not be foreseen, and generally would affect one or a few process instances [31].
– **Dynamism**: the ability of the workflow process to change when the business process evolves. This evolution may be slight as for process improvements, or drastic as for process innovation or process reengineering [31].
– **Complexity**: the measures of the difficulty to model, analyze, and deploy a process model [7], as well as the support for the dynamic and changing business process.

Third, a functional comparison, followed by an empirical comparison is carried out. The **functional** comparison uses workflow control patterns [4] as the benchmark for examining the expressibility and complexity of the modeling languages. Workflow control patterns address the requirements for the modeling languages instead of the overall methodology to model business processes, which provide a mean to examine the expressibility of a particular process modeling language. FlowMake [30], PLM$_{flow}$ [38] and ADEPT [17] have been chosen as language representatives, where mappings from workflow patterns (including basic control flow, advanced branching and synchronization patterns [4]) to the language-specific model constructs of each language are carried out. While in the **empirical** comparison, the selected languages are used to model a real-life business process which involves complex control flow structures including multiple choices, parallel executions and indefinite looping.

Lastly, based on the results of comparisons, each language representative is analyzed according to the identified comparison criteria. The analysis results are then cross-referenced when critical remarks are given for the graph-based and rule-based modeling approach.

4 Comparative Analysis

In this section, key results for the comparative analysis are presented according to the study in [23].

When considering **design time** characteristics, graph-based languages have formal foundation in graph theory, which provides rich mathematical properties for the syntax and semantics and theoretical support. The process definition is robust and structurally sound. Besides, graph theory is well-known and has been well-studied, and most importantly, it is visual and hence intuitive and useful for all kinds of workflow designers (with or without technical background). On the other hand, while having their root in formal logic, rule-based languages are competitive to the graph-based rival in terms of mathematical soundness, model robustness and myriad of model checking techniques. However, rule-based modeling languages are inevitably more **complex**, reflected by the effort to specify, reason about and manage a large number of rules for complex business processes, which require reasonable proficiency in rule-based formalism.

In terms of **expressibility**, the richness of the graph-based language syntax allows explicit specifications for complex workflow constructs. The mapping from workflow control patterns [4] to FlowMake constructs is straightforward. The goal for graph-based process modeling is to provide a structurally and semantically correct process definition that is suitable for business process automation [32]. To ensure structural correctness all possible execution paths must be defined at design time and verified. Rule-based languages are able to represent all considered workflow control patterns, in that simple rule expressions connected by AND and OR operators are capable to express same constructs as those specified by basic and advanced graphical operators (e.g., choice, merge, fork, synchronizer, discriminator [4, 32]). Furthermore, rule-based languages are slightly more expressive than graph-based languages. An obvious example is the ability to specify the temporal requirement in addition, e.g., the relative deadline for a task execution.

When considering **runtime** characteristics such as ad hoc modification to workflow schemas and exception handling, the rule-based approach takes the advantage. The rigidity in graph-based models incurs problems of lack of **flexibility**, **dynamism** and **adaptability**, which compromise the ability of the graph-based processes to react to dynamic changes in business process and exceptional circumstances. Although there have been proposals [29, 31] to cope with such, e.g., to define a set of operation rules such that runtime modifications to current process model do not introduce any conflicts, the incorporation of change operations contributes to the overall modeling **complexity**. In the rule-based approach, the completeness requirement for process models is relaxed, which provides the ability to deploy partially-specified process definitions (in rules). This is supported by the enactment mechanism, the rule engine which performs logical inferences at runtime, i.e., to determine what to execute by evaluating relevant rules on certain process event. In addition, process logic of underlying business processes is externalized from the execution environment. As a result, runtime modifications to process definition can be realized by amending the existing set of rules (i.e., modify, insert and delete existing rules to reflect changes in process logic or to implement process improvement) without impacting the executing process instances.

Table 3 provides an overview of the above discussion.

Table 3. Summary of comparative approaches.

Criteria	Graph-Based Language	Rule-Based Language
Expressibility	– Able to express structure, data, and execution requirements. – Most examined workflow patterns [23] can be expressed by the graph-based representative.	– Able to express structure, data, execution, as well as temporal requirements. – Rule expressions can represent more workflow patterns than graph-based languages.
Flexibility	– Processes can only be executed on complete process models, in which all possible execution scenarios are explicitly specified. The conditions for each scenario must also be articulated in the process model a priori.	– More flexible as incomplete specification for task dependency is supported, e.g., in ADEPT [17], it is possible to specify *task T2 and T3 must execute after T1 has finished, either in parallel or serially* in a single rule expression without explicit specification on the conditions for parallel or serial execution.
Adaptability	– Exceptions rise if some executional behavior occurred that has not been defined in the process model. Exception handlings need to be defined through an additional set of policies (rules).	– Anticipated exceptions can be handled by specifying additional rules besides that for expressing regular process logic.
Dynamism	– Support for ad hoc changes to process model after deployment is limited. – Requires defining a complete and minimal set of change operations in order to preserve the structural dependencies of the process model and running process instances.	– Rule expressions can be revised at runtime to realize ad hoc changes to process logic. – Impact on running process instances is minimal as process logic is isolated from the executional environment. – Process refinement can be rapidly implemented.
Complexity	– Modeling languages have more abstract syntax and simpler semantics, thus less complexity in model representation and verification. – Runtime support for dynamic workflows is more complex as extra modeling constructs and verification effort is required.	– Model languages have a logical syntax and required some expertise when modeling. – Process models have no visual appeal, verification process is more complex (however can be automated by logic reasoning engine). – Change to process logic is realized by rule modification and hence less complex.

5 Process Modeling Support in Commercial Products

Industry researches show that many process modelers are business process owners rather than technical personnel [10]. Graph-based languages have the visual appeal of being intuitive and explicit, even for those who have little or no technical background. However, rule-based modeling languages such as in ADEPT and PLM$_{flow}$ require good understanding of propositional logic and the syntax of logical expressions, thus are less attractive from the usability point of view. For this reason, most commercial BPMS, as well as current industrial standard including BPMN [27] by Business Process Management Initiative (BPMI) endorse the graph-based formalism as the process definition language. Examples of many commercial BPMS that use graphical process definition include SAP NetWeaver, Tibco Staffware Process Suite [34], and Utimus [35]).

On the other hand, the rule-based approach is often referred to as business rule management systems (BRMS). The common objective of BRMS is to integrate complex process logic into process model as rules, but externalize from BPMS in order to support dynamic changes. BRMS provides well-supported functions such that rule evaluation can be performed efficiently [15]. Example BRMS include Tibco iProcess Decisions [34], and ILOG JRules [15].

Many vendors advocate that rule-driven BPMS presents best solutions under current business requirements as a compromise of graphical appeal and the power of rules. A rule-driven BPMS is a superset of BRMS and BPMS, which provides rich development options for many aspects of BPMS development, of both procedural (graph) and declarative (rule) approach. The technique to realize this is to maintain a set of business rules in the external BRMS which can automate complex decision making in BPMS (i.e., when a complex decision has to be made to choose execution path). An example can be ILOG BPM [15], which is a BPMS enhanced by the functionality of the ILOG Rule Engine. Business processes are modeled in the BPM component, while the BRM component is responsible for formulate, compile, manage, and update business rules (in the E-C-A paradigm), invoke rule evaluation, and communicate with BPM component upon request to automate decision making.

6 Summary and Future Work

We proposed the use of control flow capabilities as the starting point for evaluating business process models based on two dominant approaches, namely graph-based and rule-based. This paper has presented the result of a critical and comprehensive analysis of these two prominent modeling approaches for business processes, with the focus on control flow capabilities. The presented survey gives an overview of process modeling languages developed using these two formalisms. The analysis of two approaches reviews their strengths and weakness in terms of expressibility, flexibility, adaptability, dynamism and complexity considerations. The intended future work includes a more detail empirical comparison on a selection of typical business scenarios, and with the focus on capabilities, other than control flow, of the process modeling approaches.

References

1. van der Aalst. W. M. P.: Three Good reasons for Using a Petri-net-based Workflow Management System. In proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC'96), Cambridge, Massachusetts (1996)
2. van der Aalst, W. M. P. : Verification of Workflow Nets. In proceedings of Application and Theory of Petri Nets. Lecture Notes in Computer Science, Vol. 1248 (1997) 407 – 426
3. van der Aalst, W. M. P., ter Hofstede, A.H.M.: YAWL - Yet Another Workflow Language. Information Systems, Vol. 30(4) (2005) 245-275
4. van der Aalst, W. M. P., ter Hofstede, A.H.M. Kiepuszewski, B., Barros, A.P.: Workflow Patterns. Distributed and Parallel Databases, Vol. 14(3) (2003) 5-51
5. van der Aalst, W. M. P., t. Hofstede, A. H. M., Weske, M.: Business Process Management: A Survey. In proceedings of Conference on Business Process Management (BPM 2003), Eindhoven, The Netherlands (2003)
6. Bae, J., Bae, H., Kang, S., Kim, Y.: Automatic Control of Workflow Processes Using ECA Rules. In IEEE Transactions on Knowledge and Data Engineering, Vol. 16(8) (2004)
7. Cardoso, J.: How to Measure the Control-Flow Complexity of Web Processes and Workflows. In: Fischer, L. (ed.) The Workflow Handbook, WfMC (2005) 199-212
8. Casati, F., Ceri, S., Pernici, B., Pozzi, G.: Conceptual Modeling of Workflows. In Proceedings of 14th International Conference of Object-Oriented and Entity-Relationship Modeling (OOER'95), Gold Coast, Australia (1995)
9. Casati, F., Grefen, P., Pernici, B., Pozzi, G., Sánchez, G.: A Specification Language for the WIDE Workflow Model. Technical report, University of Twente (1996)
10. Delphi Group: BPM 2005 Market Milestone Report. (2005) URL: http://www.delphigroup.com/research/whitepapers.htm.
11. Ellis, C. A., Nutt, G. J.: Modeling and Enactment of Workflow Systems. in Proceedings of 14th International Conference of Application and Theory of Petri Nets, Chicago, USA (1993)
12. Green, P., Rosemann, M, Indulska, M.: Ontological Evaluation of Enterprise Systems Interoperability Using ebXML. In IEEE Transactions on Knowledge and Data Engineering, Vol. 17(5) (2005)
13. Hayes-Roth, F.: Rule-Based Systems. Communications of the ACM, Vol. 28(9) (1985) 921–932
14. Herbst, H., Knolmayer, G., Myrach, T., Schlesinger, M.: The Specification Of Business Rules: A Comparison of Selected Methodologies. In Verrijn-Stuart A.A., Olle, T.W. (eds.) Methods and Associated Tools for the Information System Life Cycle, North-Holland, IFIP–18 (1994)
15. ILOG: ILOG Components for Business Process Management Solutions. White Paper, ILOG (2006) URL: www.ilog.com
16. Jenssens, G. K., Verelst, J., Weyn, B.: Techniques for Modeling Workflows and Their Support of Reuse. In Business Process Managements - Models, Techniques and Empirical Studies, Lecture Notes in Computer Science, Vol. 1806 (2000) 1-15
17. Jennings, N. R., Faratin, P., Norman, T. J., O'Brien, P., Odgers, B., Alty, J. L.: Implementing a Business Process Management System using ADEPT: a Real-World Case Study, International Journal of Applied Artificial Intelligence, Vol. 14 (2000) 421-463
18. Kappel, G., Rausch-Schott, S., Retschitzegger, W.: A Framework for Workflow Management Systems Based on Objects, Rules and Roles. In ACM Computing Surveys, Vol. 32 (2000)
19. Kappel, G., Rausch-Schott, S., Retschitzegger, W.: Coordination in Workflow Management Systems – a Rule-Based Approach. In Conen, W., Neumann, G. (eds.) Coordination

Technology for Collaborative Applications - Organizations, Processes, and Agents, Springer LNCS 1364 (1998) 99-120
20. Kaschek, R., Pavlov, R., Shekhovtsov, V. A., Zlatkin, S.: Towards Selecting Among Business Process Modeling Methodologies. In Proceedings of 9th International Conference on Business Information Systems (BIS2006), Klagenfurt, Austria (2006)
21. Knolmayer, G., Endl, R., Pfahrer, M.: Modeling Processes and Workflows by Business Rules. In van der Aalst W.M.P (Eds,) Business Process Management, LCNS 1806, Springer-Verlag Berlin Heidelberg (2000) 16-29
22. Liu, L., Pu, C.: ActivityFlow: Towards Incremental Specification and Flexible Coordination of Workflow Activities. In proceedings of 16th International Conference on Conceptual Modeling / the Entity Relationship Approach (ER 97), Los Angeles, USA (1997) 169-182
23. Lu, R.: Comparison of Workflow Modeling Approaches. Honors Thesis, School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane (2004)
24. Merz, M., Moldt, D., Muller K., Lamersdorf, W.: Workflow Modeling and Execution with Colored Petri Nets in COSM. In proceedings of the Workshop on Applications of Petri Nets to Protocols within the 16th International Conference on Application and Theory of Petri Nets (1995) 1-12
25. Müller, R., Greiner, U., Rahm, Erhard.: AgentWork: a Workflow System Supporting Rule-Based Workflow Adaptation. In Data & Knowledge Engineering, Vol. 51(2) (2004) 223-256
26. OASIS: Business Process Execution Language for Web Services Version 1.1 (BPEL4WS 1.1) Specification. (2006)
27. Object Management Group: Business Process Modeling Notation (BPMN) Specification 1.0 (2006)
28. Oracle: Building Flexible Enterprise Processes Using Oracle Business Rules and BPEL Process Manager, An Oracle White Paper, Oracle (2005) URL: www.oracle.com
29. Reichert, M., Dadam, P.: $\text{ADEPT}_{\text{flex}}$ - Supporting Dynamic Changes of Workflows without Losing Control. Journal of Intelligent Information Systems, Special Issue on Workflow Management, Vol. 10 (1998) 93-129
30. Sadiq W., Orlowska, M.: On Capturing Process Requirements of Workflow Based Business Information System. In proceedings of 3rd International Conference on Business Information Systems (BIS '99), Poznan, Poland (1999)
31. Sadiq, S., Sadiq, W., Orlowska, M.: A Framework for Constraint Specification and Validation in Flexible Workflows. Information Systems, Vol. 30(5) (2005)
32. Sadiq, W., Orlowska, M.: On Correctness Issues in Conceptual Modeling of Workflows. In proceedings of European Conference on Information Systems (ECIS '97), Cork, Ireland (1997)
33. Sedera, W., Rosemann, M, Doebeli, G.: A Process Modeling Success Model: Insights from a Case Study. In Ciborra CU, Mercurio R, de Marco M, Martinez M, Carignani A (eds.), Proceedings of the 18^{th} European Conference on Information Systems, Naples, Italy (2003)
34. Tibco: Enhancing BPM with a Business Rule Engine, Tibco White Paper, Tibco, (2006) URL: http://www.tibco.com
35. Ultimus: Adaptive Discovery: Accelerating the Deployment and Adaptation of Automated Business Processes. White Paper, Ultimus Inc. (2004) URL: www.ultimus.com
36. Wikarski, D.: An Introduction to Modular Process Nets. Technical Report TR-96-019 International Computer Science Institute (ICSI), Berkeley, CA, USA (1996)
37. Workflow Management Coalition: Workflow Process Definition Interface - XML Process Definition Language, (2006) URL: http://www.wfmc.org/standards/docs.htm

38. Zeng, L., Flaxer, D., Chang, H., Jeng, J.: $\mathrm{PLM}_{\mathrm{flow}}$: Dynamic Business Process Composition and Execution by Rule Inference. In proceedings of 3rd VLDB Workshop on Technologies for E-Services (TES'02), Hong Kong, China (2002)
39. Zeng, L., Ngu, A., Benatallah, B., O'Dell, M.: An Agent-Based Approach for Supporting Cross-Enterprise Workflows. In proceedings of the 12th Australasian Database Conference (ADC2001) (2001)
40. Zisman, M. D.: Representation, Specification and Automation of Office Procedures. PhD Thesis, Wharton School of Business, University of Pennsylvania (1977)