Why do we actually need the Pi-Calculus for Business Process Management?

Frank Puhlmann

Business Process Technology Group
Hasso-Plattner-Institute for IT Systems Engineering
at the University of Potsdam
D-14482 Potsdam, Germany
frank.puhlmann@hpi.uni-potsdam.de

Abstract. This paper discusses the applicability of a process algebra, the π-calculus, as a formal foundation for Business Process Management (BPM). We therefore investigate the π-calculus from a technical viewpoint based on current work in this area. The paper summarizes shifting requirements and discusses evolving theories behind BPM from the past over state–of–the–art to the future. The concepts and theories are concluded by an illustrating example highlighting why the π-calculus is a promising foundation for future BPM.

1 Introduction

This paper discusses the applicability of a process algebra for the description of mobile systems, the π-calculus [1], as a formal foundation for *Business Process Management* (BPM). BPM refers to an integrated set of activities for designing, enacting, managing, analyzing, optimizing, and adapting *business processes*. To distinguish economical from technical aspects, we denote computerized business processes as *workflows*. Still, BPM lacks a uniform formal foundation for the technical aspects [2], that might be filled by the π-calculus.

The paper complements the work on the π-calculus as a foundation for BPM from an economical viewpoint as investigated by Smith and Fingar [3, 4] by focusing on technical aspects. We therefore analyze three major shifts in the area of BPM, ranging from system description and distribution aspects up to changing environments in which business processes are executed. After describing these shifting requirements, we analyze how the theory of computer science can pace up with them starting by sequential over parallel up to mobile system theories. We argue that the arising requirements for BPM can only be fulfilled by mobile system theory, such as the π-calculus. Mobile systems are a complete new approach to BPM, that have not yet been investigated as a formal foundation for BPM. We discuss concepts like Workflow Patterns [5], service orchestration and choreography as well as correlations [6] using recent work in the area of mobile systems and BPM [7, 8].

The paper is organized as follows. We first introduce the shifting requirements for BPM in section 2, followed by the discussion of theories for representing them

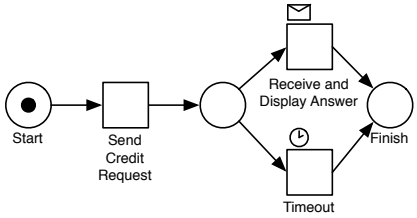


Fig. 1. Sample workflow in Workflow net notation [9].

in section 3. The paper continues with an example in section 4 and concludes in section 5.

2 Shifting Requirements

Today there is an emerging shift in the area of Business Process Management. People are used to state-based Workflow Management Systems (WfMS). In these, a workflow consists of several activities or tasks[1] guided by explicit control flow that defines the state the workflow is in [9]. The workflow itself often resembles some kind of office process that is enacted in a well defined and closed environment like the department of an enterprise. Structural change occurs seldom and can be handled in a pre-defined manner [10]. An example is given in figure 1.

However, things are changing to more flexible, distributed workflows. These "new" workflows still have a state and are guided by control flow constraints. But the "new state" is made up of events instead of documents in certain places. These events are consumed and produced by activities that have no static connections but event-connectors. Events are used as preconditions to trigger activities and produced as an outcome. Some activities are still executed in closed environments, but most are distributed as services over open environments like the internet. There is no absolute control of the workflow by a single engine, or by any engine, as activities are outsourced and fulfilled not by technical issues but rather by legal contracts. The event-based model allows the flexible integration of activities and sub-workflows into other workflows, wherever they are distributed, and however they are executed, as long as their interaction behavior is matching.

These sketched shifts in BPM bring up some very interesting questions regarding the formal, technical representation that will be discussed in detail now. Each subsection is concluded with a short paragraph summarizing the issues and deriving requirements for formalisms.

[1] Although sometimes used differently, we use activity and task synonymous in this paper.

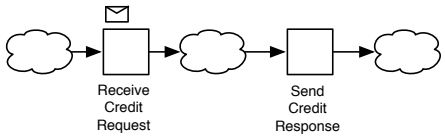


Fig. 2. An abstract process for interaction with the workflow shown in figure 1.

2.1 From State to Message-based Systems

Traditional, academic workflow research focuses on state-based descriptions of workflow, e.g. Workflow nets [9]. However, as the term workflow broadens to inter-organizational workflow between companies, state-based process descriptions have come to their limitations. This is especially true within service-oriented computing (SOC) containing orchestration and choreography as a realization for intra- and inter-organizational workflow [6].

To underpin these assumption, let's take a closer look at the example shown in figure 1. The simple workflow consists of a task that sends a credit request and afterwards waits for either the response or a timeout if no response has been received within a given timeframe. In traditional workflow systems, this process lives alone. Each task appears at the work list of some employee who finally executes it. The first task consists of writing and sending a letter. Afterwards, two exclusive tasks appear at the work list. If an answer is received by mail within the given timeframe, the answer is processed, whereas otherwise the timeout task is selected (which contains some fallback actions).

Future BPM incorporates the service-oriented computing paradigm, where now all tasks of the example are managed by computers. But what is required for this to take place? Of course, a corresponding process we can interact with. Let us assume this to be an abstract process, meaning that we only know the parts we can use for interaction, shown in figure 2.

We use clouds to denote the hidden parts of the corresponding process and call it a service. All we know about it is the interface description (receive a request, send response with the corresponding parameter format not shown in the visualization), as well as the interaction behavior (first receive a request, then send a response). To denote the interaction between our workflow and the service as a state-based description, we need to introduce additional states that describe incoming requests and outgoing responses. The result contains two Workflow nets which interact by shared places, shown in figure 3.

We want to stress that the complete system consists of two different workflows, executed by two different engines that use shared places for synchronization. There exists some theoretical work of how to extend the service without violating the interface behavior [11], as well as how to use state-based systems to formalize these kinds of workflows systems, e.g. [12, 13].

Message-based systems, in contrast, have no need for this artificial layer of virtual places to tie together workflow interactions. In practice, all state-based

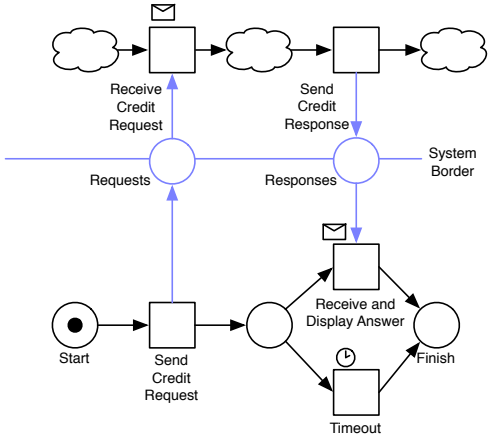


Fig. 3. State-based interaction between the workflows from figure 1 and 2.

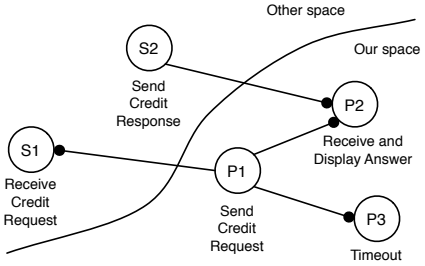


Fig. 4. Message-based routing and interaction view of figure 3.

workflow engines rely on message exchange for synchronizing their workflows. The requests and responses of our sample workflow are actually messages that move around, i.e. a special kind of event. But why should we use formalisms relying on states and implementations based partly on states and partly on messages? What if we not only base the interaction between different workflows on messagess, but also the routing of the workflow itself?

The idea of routing workflows based on events or messages has been adapted from active database management systems [14]. Each task has preconditions that have to yield true to enable the task, and postconditions that hold after the task has been executed. An example is shown in figure 4.

All tasks are represented as circles with a short name inside (not to be confused with the states, or places of Workflow nets). The pre- and postconditions of the tasks are not shown in the figure, however the dependencies between are denoted by lines. Each line connects a postcondition of one task with the precondition of another one, where the precondition-end is marked with a filled circle.

We can detect dependencies between $P1$ and $S1$, $P2$, $P3$ as well as $S2$ and $P2$. The tasks $S1$, $P2$, and $P3$ can only be activated after $P1$ has been executed, thus meaning the preconditions of $S1$, $P2$, and $P3$ depend on the postconditions of $P1$. The task $S2$ has some preconditions linked to $S1$ that are not known to us.

As a key result of the shift to message-based systems, we can denote intra- as well as inter-organizational workflow in one message-based paradigm for implementation and formalization.

BPM Shift I: BPM shifts from state to message-based systems as the later supports intra- and inter-organizational workflows without a paradigm break between formalism and practical implementation.

New Requirement I: A formalism representing the first shift of BPM should be based on messages, or events, rather than states.

2.2 From Central Engines to Distributed Services

As BPM Shift I is the foundation for the other shifts, figure 4 already denotes the next shift moving from central engines to distributed services. The message-based paradigm described in the last section not only describes how tasks in a workflow can be routed and interaction between different workflows can be modeled but furthermore introduces a loose coupling between the tasks.

These loose coupling between different tasks allows for highly distributed systems. Instead of having a single engine controlling every aspect of a workflow, this knowledge is now spread across the parts of collaborating workflows. Distributed tasks, called services, wait for messages to arrive that trigger the activation and produce new messages to trigger other tasks. Of course, there are some distinctions to be made.

They start with different spaces in the environment for the distributed tasks to live in. In figure 4, this is denoted with *Our Space* and *Other Space*. Our space is usually something like an intranet, where we control things like access conditions, availability, implementation issues and so on. We make some of our services available to the outer world, acting as interaction points, without providing knowledge or access to our internal structures. Indeed, we are free to restructure our internals as wanted. Our workflow incorporates other tasks that are available in the other space, typically in other intranets or the internet. These other tasks are parts of systems such as ours, and represent interaction interfaces. However, we have only limited control over them, mostly by legal agreements. We cannot enforce their availability, functionality, or implementation. Still, we are free to drop them as interaction partners and bound to others. This high flexibility requires the shift from closed, central engines to open, distributed services for representing workflows.

BPM Shift II: BPM shifts from central engines to distributed systems as integration becomes a core BPM activity.

New Requirement II: A formalism supporting the second shift of BPM should support advanced composition and visibility of its parts.

2.3 From Closed to Open Environments

In traditional workflow theory, the execution environments are quite static and foreseeable. We denote this kind of environment closed, as the level of external influences is rather low. However, with shifts from traditional departments and companies up to virtual organizations and agile, customer order-specific collaborations, the execution environment is shifting, too. This new kind of environment is called open and is represented by large intranets as well as the internet.

Closed environments are usually accessible, deterministic, quite static, and have a limited number of possible actions. Accessibility describes the level of knowledge we have about the execution environment regarding completeness, accuracy, and up-to-dateness of the information. In a single company or department we should be able to exactly capture this knowledge. If we expand the environment to the whole internet, there is much left in the dark that could be useful or crucial for our business, however we are simply unable to find and incorporate it.

Executing a task in an open environment is more uncertain then in a closed one. This is denoted as the determinism of the actions. In an open environment they are way more possibilities to foresee and handle. However, if the environment is complex enough, as e.g. the internet, we can not enforce everything.

While closed environments are quite static, open environments tend to be constantly changing in large parts, regardless of our actions. Interaction partners appear, disappear, are prevented, or something else happens that we have to take into account for our business to run.

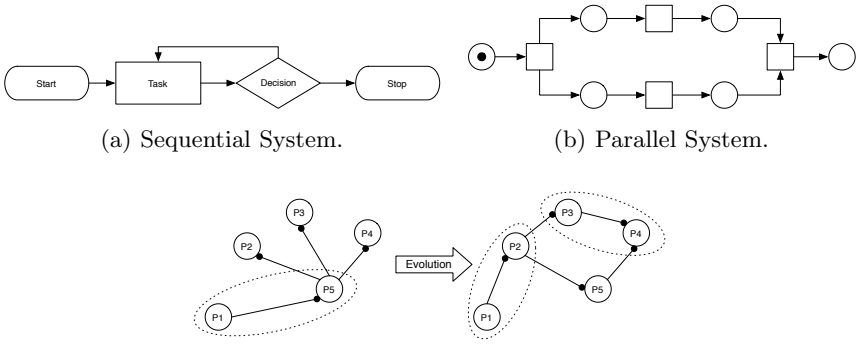
Furthermore, the number of interaction partners that can be invoked to perform a certain task is rising fast as the environment opens to the world. So our decision making process of whom we incorporate into our business is getting more complex.

BPM Shift III: BPM shifts from closed to open environments as the application area is extended from internal office workflows to agile collaborations in open networks like the internet.

New Requirement III: A formalism taking the third shift into account should support dynamic process structures that can support change.

3 Evolving Theories

This section evaluates how we can cope with the shifting requirements from a theoretical viewpoint. The theories behind computer science have evolved in several steps, starting from theories describing sequential over parallel up to mobile systems. Reasons why a theoretical foundation for BPM is required are



(a) Sequential System. (b) Parallel System.

(c) Mobile System.

Fig. 5. Theories for describing BPM systems.

manifold. Well known reasons include the unambiguity of formal models as well as the possibility for analysis. A detailed discussion can be found in [2].

Sequential systems are described in the λ-calculus by Alonzo Church and Stephen Cole Kleene as well as Turing machines from Alan Turing. Both theories have been developed in the 1930s. The science evolved to the description of parallel systems by the use of Petri nets, developed by Carl Adam Petri in the 1960s. Another thirty years later, at around 1990, a theory of mobile systems, called the π-calculus, has been developed by Robin Milner, Joachim Parrow, and David Walker. Sequential systems are based on the concept of executing one step after the other. Parallel systems as represented by Petri nets explicitly describe static structures for concurrently executed steps. Mobile systems are made up of parallel components which communicate and change their structure – thereby overcoming the limitations of static structures.

The following subsections discuss each of these theories regarding to the shifting BPM requirements.

3.1 Sequential Systems

Sequential systems can be formally described by the λ-calculus [15] as well as Turing machines [16]. The λ-calculus is a formal system designed to investigate the definition of functions, which are used for sequential computing. It brought the ideas of recursion and the precise definition of a computable function into discussion even before the first computers have been constructed. In the view of computer science, the λ-calculus can be seen as the smallest universal programming language as any computable function can be expressed and evaluated using this formalism. A different approach, computational equal to the λ-calculus, are Turing machines that form the foundation for imperative programming languages. Both, the λ-calculus as well as Turing machines, can be used to represent business processes at a very low level of abstraction, already making concepts

like parallelism a complexity overwhelm. A typical representation of a sequential system are flow-charts, represented in figure 5(a).

BPM and Sequential Systems. Sequential systems are widely used for everyday BPM implementations. They can be used to model messages, distribution, as well as dynamic process structures at a very low level of abstraction. However, the low abstraction of their respective formalisms make them complexity overwhelming as a theoretical foundation for modeling and reasoning about business processes.

3.2 Parallel Systems

While the λ-calculus formed the foundation for many computer science related topics like programming languages, the description of workflows required a different approach. In a typical workflow tasks are not only executed in sequential order, furthermore tasks are executed in parallel by different employees to speed up the processing. These different – then again sequential – processing paths have to be created and joined at some points in the business process. Even further, parallel processing tasks could depend on each other. The optimization of business processes usually adds parallelism and dependencies as this is an effective way to reduce the throughput time.

These kinds of parallel processes are hard to describe in terms of the λ-calculus. To overcome the limitations of sequential systems, an approach to represent parallel systems called Petri nets [17] has been adapted for workflow representation. Petri nets have a simple but yet powerful mathematical foundation as well as a strong visual representation. They use the concept of an explicit state representation for parallel systems. Each Petri net is always in a precisely defined state denoted by the distribution of tokens over places contained in the net. The state of the system could then be changed by firing transitions which relocate the token distribution over the places. Petri nets have been adapted by many systems that are used in the business process management domain to describe business processes [18, 19]. A Petri net, as a typical representation of a parallel system, is shown in figure 5(b).

Beside the advantages of Petri nets for the business process management domain, that include strong visualization capabilities, mathematical foundations, as well as their main purpose, the description of parallel systems, Petri nets also have some drawbacks. The main drawbacks are the static structure of the nets (that do not support dynamic process structures) as well as the missing capabilities for advanced composition as for instance recursion. Of course, Petri have been extended with support for dynamic structure, like self modifying Petri nets [20], recursion [21], and objects [22]. However, these enhancements also complicate the theory of the nets and thus have reached restricted usage only. A broad research on the capabilities of Petri nets regarding common patterns of behavior found in business processes showed that they fulfill basic tasks like splitting and merging process paths easily, while they fail at advanced patterns like multiple instances of a task with dynamic boundaries [5]. Whereas there exist

approaches to overcome some or all of the limitations regarding the behavior [9, 23], the static structure and limited composition of Petri nets remains.

BPM and Parallel Systems. Parallel systems, i.e. Petri nets, are quite common in modeling workflows at a rather high abstraction level. They allow a formalized reasoning on things like deadlocks, livelocks, or optimization issues. However, they fail to deliver a robust foundation even for common workflow patterns [23], as well as supporting dynamic process structures. Distribution and messages required for BPM systems are partly possible, but add more complexity without directly supporting the underlaying concepts.

3.3 Mobile Systems

To overcome the limitations of Petri nets, which are not only concerning the BPM domain, theories of mobile systems have been developed. Thereby a mobile system is made up of entities that move in a certain space. The space consists of processes, and the entities that move are either links between the processes (link passing mobility) or the processes themselves (process passing mobility).

A theory for mobile systems, the π-calculus, overcomes the limitations of Petri nets regarding the static structure and limited composition capabilities at the cost of a more complex representation. The π-calculus represents mobility by directly expressing movements of links in an abstract space of linked processes (i.e. link passing mobility). Practical examples are hypertext links that can be created, passed around, and disappear. The π-calculus does not, however, support another kind of mobility that represents the movement of processes. An example is code that is sent across a network and executed at its destination.

The π-calculus uses the concept of *names* with a certain scope for interaction between different parallel processes. Names are a collective term for concepts like channels, links, pointers, and so on. As the mobile system evolves, names are communicated between processes and extrude or intrude their scope regarding to certain processes. As the synchronization between processes is based on interaction and received names are also used as communication channels, the link structure is changed dynamically all the time the mobile system evolves. An example is shown in figure 5(c). The processes are denoted as circles, whereas the links between are lines. A dot at a line end represents a process that is receiving messages on this link, the scope (visibility) of a link is denoted with a dotted circle. As the link structure of mobile systems changes all the time the processes evolve, we can only draw snapshots of the current system configuration. This is denoted with the *Evolution* arrow in the figure.

BPM and Mobile Systems. Mobile Systems, i.e. the π-calculus, are recently discussed as foundations for BPM, see e.g. [3, 4, 24]. Standards like WS-CDL [25] or BPML [26] claim to be based on the π-calculus. Recent research has shown that the π-calculus is indeed able to easily implement all of the documented workflow patterns [7] as well as represent powerful service choreographies [8]. The π-calculus allows the direct representation of message-based systems, with the

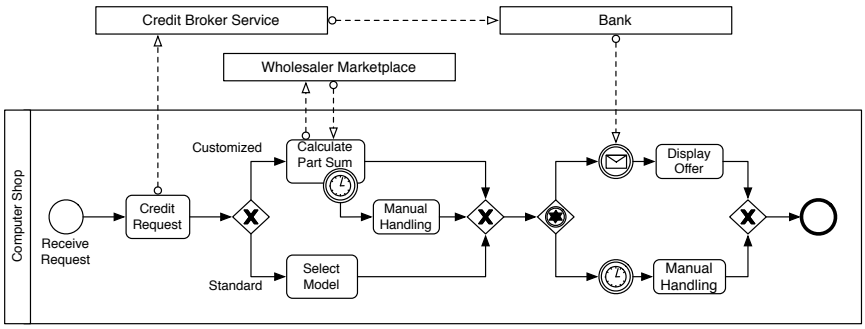


Fig. 6. A Workflow/Collaboration view in BPMN notation [27].

ability of distributing its components. Link mobility supports dynamic, evolving process structures within the core grammar of the calculus. It is already used to describe service correlations in WS-CDL, formally described in [8]. Together with a message-based representation, link mobility can also be used to dynamically include new behavioral as well as interaction patterns in running workflows.

4 Example

This section investigates the shifting requirements for BPM based on a practical example and discusses the pros and cons of different formalizations.

Figure 6 shows an internal workflow of a computer shop as well as its interaction with the (digital) environment. We use the Business Process Modeling Notation (BPMN), as it is the only standard notation supporting orchestration as well as choreography [27].

The computer shop offers the advantage of financing a computer purchase with the lowest possible interest for a certain customer as well as calculating realtime prices for the components. For the first advantage, the shop incorporates an external credit broker service. The services mediates loans up to €5000 based on the personal data of the customer. After finding the lowest customer specific interest, the offering bank is informed to send a binding credit offer to the computer shop. The second advantage is given by incorporating a wholesaler marketplace that searches for the lowest available component prices.

The example in figure 6 contains the internal workflow for processing a request from a customer. This results in an offer containing a loan offer with the specific conditions. As the credit processing takes some time and is independent of a specific sum up to €5000, it is the first task in the internal workflow. Technically, it is an asynchronous service invocation, containing a correlation to the workflow instance for the callback action of the bank. Afterwards, the customer request is further evaluated. If the customer selects a standard, pre-configured computer, the price is taken from an internal database. If the computer should be customized, the wholesaler marketplace is searched for the lowest available

prices. Technically, this is implemented by a synchronous service invocation. If this task did not finish within a given timeframe, the offer has to be handled manually. Afterward the workflow consists of a deferred choice pattern, modeled using an event-based BPMN gateway. If a response from a bank is received, the offer is displayed. Otherwise a manual handling is required.

The interesting questions in this example, from a theoretical as well as practical viewpoint, are then: How can we represent the internal workflow matching to the choreography? How do we ensure that the selected bank can call back exactly our instance of the workflow, while there are many banks for the credit broker service to choose from? Further problems, that are not discussed here, are later on for example: How do we interact with a bank maybe not known before? How can we incorporate the interaction pattern dynamically in our workflow (e.g. requesting a security, further information, etc.)?

For having a precise understanding of the processes described, the graphical notation, as well as possible assumptions, have to be formalized. The formalization can then be used to enact the workflow as well as allow reasoning about properties like soundness, interaction correctness, or non-functional properties. An extended example can for instance use a more complex interaction pattern for the wholesaler marketplace, including callback actions if a price drops below a limit, etc. Using interaction correctness criteria, it can be checked if the internal workflow matches the interaction behavior of the marketplace.

Sequential Systems. A formalization in sequential systems is very difficult. We have different interaction partners as well as complex workflow patterns. The resulting representation, e.g. Java, would be unusable for reasoning about the properties of the processes.

Parallel Systems. A formalization with Workflow nets is partly possible. However, advanced workflow patterns, like the intermediate timeout at a certain task (called *Event-based Rerouting* [8]), are difficult to describe in Workflow nets. Sophisticated aspects like the correlation of the credit broker request with a certain instance of the workflow are even harder, resulting in very complex nets. Taking into account that we have a large set of banks, wholesalers, etc. with static connections to our workflow makes distribution quite difficult. Adapting new interaction behavior for a certain bank into our workflow requires the definition of new nets. Thus, the static structure of the nets make distribution and adaption to a changing environment quite difficult.

Mobile Systems. Mobile systems are the most promising approach of formalizing processes based on the shifting requirements. They support message-based interaction that directly complies with the distributed nature of the example, as well as the dynamic integration of initially unknown participants, like the banks. They easily describe correlations in a formal manner, and are open for the integration of new processes, as they are not linked statically. Various types of reasoning have been developed for the π-calculus. For instance, weak open

bisimulation [28] allows for matching the interaction behavior of a process with another one. A complete formalization of the example can be found in [8].

5 Conclusion

This paper investigated why traditional formalisms for workflow, like Workflow nets, are not well suited for future BPM, as they do not match the shifting requirements of message-based, distributed, and dynamically adapting processes.

Recent results [7, 8] have shown that the π-calculus is well suited for modeling classical workflows, nowadays known as service orchestrations, as well as service choreographies that together form a core foundation of future BPM based on the shifting requirements. While recent standards like BPEL4WS [29], BPMN [27], or WS-CDL [25] tackle the problem from a practical side, a formal foundation for BPM is still missing. This paper revealed the strengths of mobile systems, i.e. the π-calculus, as a formal foundation for BPM.

Acknowledgments. The author would like to thank the members of the business process technology group at the Hasso-Plattner-Institute for their interesting discussions and challenging questions regarding the π-calculus and BPM.

References

1. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes, Part I/II. Information and Computation **100** (1992) 1–77
2. van der Aalst, W.M.P., ter Hofstede, A.H., Weske, M.: Business Process Management: A Survey. In van der Aalst, W.M.P., ter Hofstede, A.H., Weske, M., eds.: Proceedings of the 1st International Conference on Business Process Management, volume 2678 of LNCS, Berlin, Springer-Verlag (2003) 1–12
3. Smith, H., Fingar, P.: Business Process Management – The Third Wave. Meghan-Kiffer Press, Tampa (2002)
4. Smith, H., Fingar, P.: Workflow is just a pi process. BPTrends (2004) `http://www.bpmi.org/downloads/LIB-2004-01-4.pdf` (September 23, 2005).
5. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.: Workflow patterns. Distributed and Parallel Databases **14**(1) (2003) 5–51
6. Havey, M.: Essential Business Process Modeling. O'Reilly, Cambridge (2005)
7. Puhlmann, F., Weske, M.: Using the pi-calculus for formalizing workflow patterns. In van der Aalst, W., Benatallah, B., Casati, F., eds.: Proceedings of the 3rd International Conference on Business Process Management, volume 3649 of LNCS, Berlin, Springer-Verlag (2005) 153–168
8. Overdick, H., Puhlmann, F., Weske, M.: Towards a formal model for agile service discovery and integration. In: International Workshop on Dynamic Web Processes, associated with the 3rd International Conference on Service Oriented Computing, Amsterdam, The Netherlands (2005)
9. van der Aalst, W., van Hee, K.: Workflow Management. MIT Press (2002)
10. van der Aalst, W.M.P.: Exterminating the Dynamic Change Bug: A Concrete Approach to Support Workflow Change. Information System Frontiers **3**(3) (2001) 297–317

11. Aalst, W.v.d., Basten, T.: Inheritance of Workflows: An approach to tackling problems related to change. Computing science reports 99/06, Eindhoven University of Technology, Eindhoven (1999)
12. van der Aalst, W.M.P., Weske, M.: The P2P approach to Interorganizational Workflow. In Dittrich, K., Geppert, A., Norrie, M., eds.: Proceedings of the 13th International Conference on Advanced Information Systems Engineering (CAiSE'01), volume 2068 of LNCS, Berlin, Springer-Verlag (2001) 140–156
13. Martens, A.: Analyzing web service based business processes. In Cerioli, M., ed.: Proceedings of Intl. Conference on Fundamental Approaches to Software Engineering (FASE'05), Part of the 2005 European Joint Conferences on Theory and Practice of Software (ETAPS'05). Volume 3442 of Lecture Notes in Computer Science., Springer-Verlag (2005)
14. Dayal, U., Hsu, M., Ladin, R.: Organizing long-running activities with triggers and transactions. In: Proceedings of the 1990 ACM SIGMOD international conference on Management of data, New York, ACM Press (1990) 204–214
15. Barendregt, H.P.: The Lambda Calculus. Elsevier, Amsterdam (1985)
16. Turing, A.M.: On computable numbers, with an application to the entscheidungsproblem. Proceedings of the London Mathematical Society **2**(42) (1936) 230–265
17. Petri, C.A.: Kommunikation mit Automaten. PhD thesis, Institut für Instrumentelle Mathematik, Bonn (1962)
18. Emmerich, W., Gruhn, V.: Funsoft nets: a petri-net based software process modeling language. In: IWSSD '91: Proceedings of the 6th international workshop on Software specification and design, Los Alamitos, CA, USA, IEEE Computer Society Press (1991) 175–184
19. van der Aalst, W.M.P.: The Application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers **8**(1) (1998) 21–66
20. Valk, R.: Self-Modifying Nets, a Natural Extension of Petri Nets. In Ausiello, G., Böhm, C., eds.: Automate, Languages, and Programming, volume 62 of LNCS, Berlin, Springer-Verlag (1978) 464–476
21. Haddad, S., Poitrenaud, D.: Theoretical aspects of recursive petri nets. In Donatelli, S., Kleijn, J., eds.: ICATPN'99, volume 1639 of LNCS, Berlin, Springer-Verlag (1999) 228–247
22. Moldt, D., Valk, R.: Object oriented petri nets in business process modeling. In van der Aalst et al., W., ed.: Business Process Management, volume 1806 of LNCS, Berlin, Springer-Verlag (2000) 254–273
23. van der Aalst, W.M.P., ter Hofstede, A.H.M.: YAWL: Yet Another Workflow Language (Revised version. Technical Report FIT-TR-2003-04, Queensland University of Technology, Brisbane (2003)
24. van der Aalst, W.M.P.: Pi calculus versus petri nets: Let us eat "humble pie" rather than further inflate the "pi hype". BPTrends (2005) `http://www.bptrends.com/publicationfiles/05%2D05%20Pi%20Calc%20vs%20Petri%20Nets%20%2D%20Aalst%2Epdf` (September 23, 2005).
25. W3C.org: Web Service Choreography Description Language. 1.0 edn. (2004)
26. BPMI.org: Business Process Modeling Language. Technical report (2002)
27. BPMI.org: Business Process Modeling Notation. 1.0 edn. (2004)
28. Sangiorgi, D.: A theory of bisimulation for the pi-calculus. In: CONCUR '93: Proceedings of the 4th International Conference on Concurrency Theory, Berlin, Springer-Verlag (1993) 127–142
29. BEA Systems, IBM, Microsoft, SAP, Siebel Systems: Business Process Execution Language for Web Services Version 1.1. (2003)