

A Role-Based Approach To Business Process Management

Keith Harrison-Broninski

Role Modellers Ltd

Brazenhall, Horn Street, Nunney

Somerset BA11 4NP, UK

khb@rolemodellers.com

Abstract - *Business Process Management (BPM) creates a new layer in IT architecture. Older integration technologies act as middleware (they sit below the applications that they connect) but BPM sits above them. Since a BPM system integrates and organises IT assets from the users' point of view, its basic capabilities should cater naturally for human collaboration. This requires three fundamental capabilities:*

1. *Flexible process topology*
2. *Event-driven and declarative activity management*
3. *A collaborative perspective*

We consider how the XML standards proposed for BPM measure up against these criteria, and hence to what degree they support collaboration. We conclude that while BPML and BPEL4WS provide the required flexibility, none of them deliver the other 2 capabilities. We then present an alternative formal system for BPM, based on Role Activity Theory, an established formal framework for describing human and organizational collaboration. This system is defined by the XML language Playwright, and supported by the BPM engine RADRunner.

Keywords: Business Process Management, Collaborative Systems, Groupware, Role Activity Theory, Role Activity Diagrams, RADs, BPML, BPEL, BPEL4WS, XPDL, ebXML, Playwright, RADRunner

1 Introduction

A number of computer technologies and business methodologies have recently converged to form a new approach to the use of IT in business operations, *Business Process Management* (BPM, [19], [20], [21]). BPM proponents claim its advantages include:

1. Reuse of existing technology and business assets.
2. Natural extension over the Internet for automated collaboration with trading partners.
3. Separation of process definitions from connector technologies for business agility.

A notable feature of BPM is that it creates a new layer in the IT architecture. Older integration technologies act as middleware - that is, they sit below the applications that they connect - but the new process layer sits above them. For example, in the e4 methodology of Computer Sciences Corporation [23], BPM sits immediately below the user access layer. BPM offers 'human-side' integration as opposed to 'machine-side' integration. This is a significant shift, which offers an opportunity to do more than bind users into the scripted and automated world of the machine. Since a BPM system integrates and organises IT assets from the users' point of view, its basic capabilities should correspond closely to the business needs and practices of the organization(s) for which it is implemented.

Hence, a BPM system must cater naturally for human collaboration. In our view, such a system requires three fundamental capabilities:

1. *Flexible process topology.* Humans are most valuable in loosely-structured, creative, exploratory activities - and such activities routinely redefine the subsequent process. BPM systems should therefore be 'mobile' as defined by the pi-calculus [18], in that process activities and interactions can cause change to the process topology.
2. *Event-driven and declarative activity management.* Humans do not sequence their activities in the manner of a procedural computer program - they take action in response to interactions with others, and changes in the state of resources to which they have access. In formal terms, human activities are event-driven (triggered by outside intervention) and declarative (enabled and validated by logical conditions).
3. *A collaborative perspective.* In automated processes, the ultimate aim is to render the distribution of data, logic and control arbitrary - machine-side integration sets out to create a single virtual space in which the real location of these things can be altered at will without impact on the process participants. However, in collaborative systems the situation is the opposite. A human process creates meaningful connections between participants whose skills, responsibilities,

authorities and resources are quite distinct and probably very different. Collaborative technology must provide a strong representation of process participants, the roles they play and the resources they own.

In this paper we consider how the various XML standards proposed for BPM (BPML [13], BPEL4WS [14], XPDL [15] and ebXML [16]) measure up against these criteria, and hence to what degree they support collaboration. We conclude that while BPML and BPEL4WS provide the required flexibility, none of them deliver the other 2 capabilities, since their roots all lie either in block-structured programming languages designed for mechanised processes or in notions of process unsuited to representing human collaboration

We then present an alternative formal system for BPM, based on Role Activity Theory, an established formal framework for describing human and organizational collaboration. This system is defined by the XML language Playwright, and supported by the BPM engine RADRunner [22].

1 Supporting Collaborative Processes

A business process engine must at the least offer the ability to co-ordinate person-person interactions, integrate existing systems using modern protocols, and build web-based applications without coding. Most if not all BPM tools provide this functionality. However, in the main they are designed to manage the rigid, "script once, run millions" kind of workflow processes. Even if they provide some capability for changing the script and/or reconfiguring the parties dynamically (thus providing a level of business agility not found in pure workflow systems), their capabilities are not a good match for untidy, one-off, open-ended, collaborative human activities.

This can be seen by looking at the standard XML languages currently proposed for process definition by vendors and standards organizations. These languages are all based on certain fundamental notions: that processes are formed of activities, and that runtime processes have instance-specific data [24]. Beyond that, however, the paradigms used are more or less different:

1. *Business Process Modeling Language* (BPML, [13]) is intended to be capable of representing any business process whatsoever. However, its nature is driven by the bias of the company which is driving its adoption, Intalio: towards process automation systems which are based on block-structured, object-oriented programming languages. Within BPML, recursive block structure plays a significant role in scoping issues that are relevant for declarations, definitions and process execution. Flow

control (routing) is handled entirely by block structure concepts (e.g., execute all the activities in the block sequentially).

The view of business process taken by BPML is that it is simply another type of computer program, albeit a high-level one in which humans carry out some of the actions. This works well for some process types - where the sequence of tasks is repetitive and mechanistic. However, it works poorly for processes where a high degree of human interaction and initiative is involved - such as those which involve collaboration. BPML does cater for agile or mobile processes (that can be changed and/or reconfigured on the fly), but human processes are not like programs at all. Hence human collaboration requires a different type of modelling technique from that provided by BPML.

2. *Business Process Execution Language for Web Services* (BPEL4WS, [14]) is an orchestration language for web services, derived from the combination of 2 previous language specifications (WSFL from IBM and XLANG from Microsoft). Like BPML, BPEL4WS is a block-structured programming language, if somewhat more focused. It allows recursive blocks, but restricts definitions and declarations to the top level. Within a block, graph-structured flow concepts are supported to a limited extent, constrained by inheritance from previous generation workflow software (only acyclic graphs, hence no loops; some constraints on going across block boundaries; a complicated semantics for determining whether an activity actually happens).

BPEL4WS is analogous to a subset of BPML, geared specifically towards the synchronisation of disparate computer systems via the use of web services. Hence it offers the same strengths with regard to support for automation (activity types specifically for message interchange, event handling, compensation and delay; attributes to support instance correlation, extraction of parts of messages, and locating service instances; support for transactions, utilizing the block structure context, exception handling and compensation) but suffers from the same weaknesses with regard to support for human collaboration.

3. *XML Process Definition Language* (XPDL, [15]) is a proposal from the Workflow Management Coalition (WfMC), an organization whose mission is to "Increase the value of customers' investment in workflow technology, decrease the risk of using workflow products, and expand the workflow market through increasing awareness for workflow" (www.wfmc.org). As such, XPDL takes an approach predating BPM, inherited from workflow systems based on Petri Nets, which have a fixed connection structure. XPDL is a graph-structured language with additional concepts to handle blocks

(scoping issues are relevant at the package and process levels). Routing is handled by specification of transitions between activities: the activities in a process can be thought of as the nodes of a directed graph, with the transitions being the edges. Conditions associated with the transitions determine at execution time which activity or activities should be executed next. An activity attribute specifies the resource(s) required to perform an activity (an expression, evaluated at execution time, which determines the resource required) and the application(s) required to implement an activity. These concepts together support the notion of a resource (e.g., participant) in conjunction with an application performing the activity.

XPDL focuses on issues relevant to the distribution of work, and is essentially traditional workflow, with no provision for agile or mobile processes. Such adaptability is, however, essential for the support of collaborative processes. When people solve problems, design products, reengineer processes, negotiate with suppliers or resolve issues for customers, the detailed tasks cannot be scripted in advance. There is a clear purpose and a general pattern to the work, but it is one aspect of the participants' responsibilities to shape the concrete details as the process unfolds. Hence XPDL is also unsuitable for the support of human collaboration.

4. *ebXML* [16] from the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) and the Organization for the Advancement of Structured Information Standards (OASIS) is designed to allow enterprises to conduct business over the Internet using an open XML-based infrastructure, and is intended as the natural successor to EDI. It includes a component Business Process Specification Schema (BPSS) that caters for process definition, and that is roughly analogous to XPDL. In ebXML, a Business Partner is an entity that engages in Business Transactions with another Business Partner(s). The message-exchange capabilities of a Party may be described by a Collaboration-Protocol Profile (CPP). The message-exchange agreement between two Parties may be described by a Collaboration-Protocol Agreement (CPA). A CPA may be created by computing the intersection of the two Partners' CPPs. Included in the CPP and CPA are details of transport, messaging, security constraints, and bindings to a Business-Process-Specification document that contains the definition of the interactions between the two parties while engaging in a specified electronic business collaboration.

Of the standard proposals described here, ebXML is the most generic - and perhaps offers the most potential for business transformation. The International Organization for Standardization (ISO) recently approved a suite of four ebXML specifications, which may further the standard's chances of widespread take-up for business transactions over the Internet. However, like EDI before

it, ebXML is not intended for the management of human collaboration, but for the support of automated trade. It caters well for the setup and management of secure network transactions, which can be given a degree of flexibility in operation, but entirely lacks facilities for the interactions and behaviour typical of human-oriented processes.

In summary, neither BPML nor BPEL4WS are in any way geared towards collaborative activities - for instance, neither language provides any way of declaring participants or defining, in the activities, an expression, possibly based on the instance values of properties, that specifies the resource(s) required to perform the activity. XPDL does allow for such concepts to be modelled, but is a workflow-derived system that is too inflexible to support human collaboration. Finally, ebXML is not intended at all for the support of human collaboration, catering only for well-defined trading patterns established by agreement between "partner" organizations.

Relating this discussion to the breakdown above of three fundamental capabilities required to support human collaboration, we arrive at the following table.

Table 1: Support for human collaboration offered by proposed standards for XML process description

	<i>BPML</i>	<i>BPEL4WS</i>	<i>XPDL</i>	<i>ebXML</i>
<i>Flexible process topology</i>	Yes	Yes	No	No
<i>Event-driven and declarative activity management</i>	Partial	No	No	Partial
<i>A collaborative perspective</i>	No	No	No	No

A different modelling technique is required, along with a process engine to support it. In search of a suitable approach, we turn to *Role Activity Theory* (RAT, [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11]) and the accompanying graphical notation, *Role Activity Diagrams* (RADs).

2 Role Activity Theory

Originally developed in 1983, RAT has been used for business process modelling in all commercial sectors for over a decade, and is the subject today of much ongoing academic and industrial research. RADs were a major feature of the Business Process Reengineering (BPR) movement in the 1990's [10]. Although RAT and

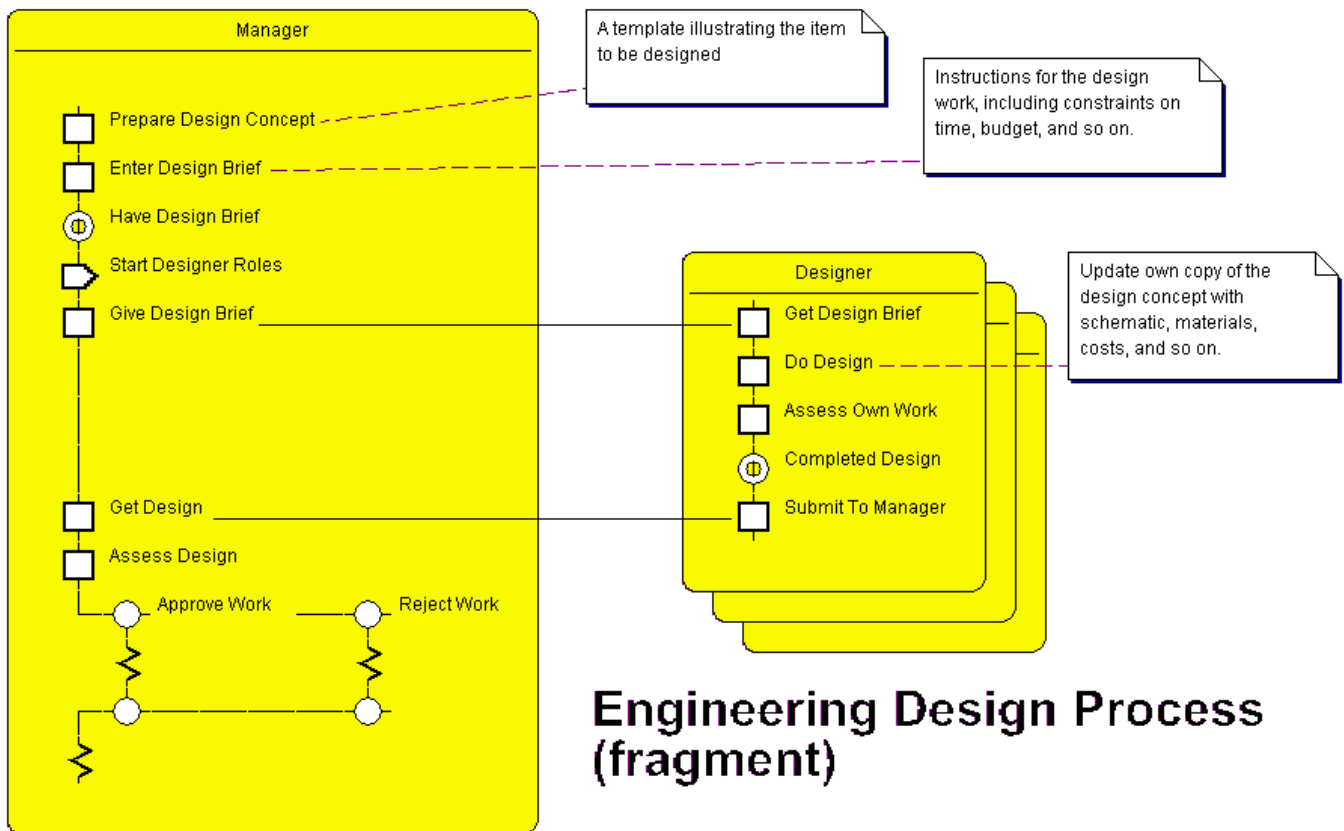


Figure 1: Example Role Activity Diagram (RAD)

RADs fell out of focus commercially as BPR lost the limelight, with the current rise of Business Process Management (BPM) and the emergence of new process-oriented software tools, there is now renewed interest in the technique.

RAT is based on the definition of 6 types of object:

1. *Users* (humans, organizations or machines)
2. *Roles* that users play within processes
3. *Resources* private to Roles, required to participate in the process
4. *Activities* carried out by Roles to manipulate resources
5. *Interactions* between Roles to transfer resources
6. *Logical conditions* to control the execution and validation of activities.

A RAD depicts:

1. The Roles played by participants within a process;
2. The information resources available to each Role;

3. The activities within each Role; and
4. How the Roles interact.

In particular, a RAD caters for flexible, dynamic activity. Rather than prescribing fixed sequences of behaviour, it is used to show precisely which information resources are needed by whom, under what conditions they become available and are required, and how these resources are used.

Although RADs have been used for many years by analysts for process modelling, their take-up for full process management has been prevented by lack of tool support. However, a BPM product based on RAT has recently become available. RADRunner (www.rolemodellers.com, [22]) is a pure J2EE application with an underlying XML dialect Playwright, which extends RAT to make provision for features required by BPM, such as automation and integration with web technologies.

3 The Approach To RAT Taken In Playwright

The original formal language for Role Activity Theory was Process Modelling Language (PML), a compiled (rather than interpreted) language developed in

the IPSE 2.5 project during the late 1980's, and later incorporated into software systems such as *ProcessWise Integrator* and *ProcessWeb* ([4], [5], [6], [7], [8], [11]). Playwright, by contrast, is an interpreted XML language. A description of PML, and comparison with Playwright, are given in the Appendix below.

Playwright and RADRunner together extend the capabilities of PML in various different ways:

- *Process management under process control*

Playwright removes from classical RAT the basis in Petri Nets, which have a fixed connection structure, with an approach implementing the pi-calculus of Robin Milner [18]. This is done by including in the language full support for users as well as Roles, in addition to built-in task types for manipulating process types and instances. Hence you can use a Role to define, start, stop, monitor, and generally manage processes and their associated users. Moreover, the interpreted nature of Playwright (as opposed to PML, a compiled language) means that changes to Role types or even instances are immediately effective - there is no need for a build-compile-deploy cycle to implement change.

- *Fully declarative process management*

Playwright activities are enabled and validated via logical conditions, not sequenced with control flow as in a workflow system or block-structured process language. Moreover, the algorithm used for automation in RADRunner can be varied in a "plug-and-play" manner - even specified individually for a particular Role instance.

- *Sophisticated control over automation*

RADRunner offers various interlocking mechanisms for semi- or fully-automated process enactment: interactive automation via a web interface, background automation via a tree of parent-child Role instances, and in-process automation control.

- *Support for collaborative web portals*

BPM is generally implemented via inclusion in a more general organizational portal. RADRunner caters naturally for this usage, since it is a thin web client application, with an interface that can be customised as desired: to include specific typefaces, images, and layout for corporate branding, stripped down for use on mobile devices, embedded as a portlet in a portal server, and so on.

- *Structured messaging*

Messaging is an enabling force for human-computer interaction, yet can result in efficiency losses as well as gains. For example, the volume of email received by organizational workers is an increasing problem - sorting it by relevance and priority alone can consume much of a working day. RADRunner permits the replacement of informal messaging with structured interactions under process control, which can take place via email, the web, or any other standard protocol.

4 Conclusions

BPM is the natural conclusion of a drive to increase automation of mundane tasks that began with the introduction of computing into the enterprise. Techniques and tools for BPM are becoming available, and adoption is increasing in the workplace. Yet BPM fails to cater naturally for an entire class of activities - those carried out by people, working together to do business in a flexible, creative fashion.

This is a serious omission. One might argue that in a world where routine processes are automated by systems, these value-adding, innovative human activities are what business is all about. An organization in which people interacted only via scripts loaded into machines could not think, could not respond, could not change and could not possibly provide effective support for its customers.

Groupware, based on email and document sharing, provides part of the answer. It facilitates collaboration but it provides no context, no understanding of what people are trying to accomplish or how they plan to go about it, no sense of what particular responsibilities or skills each participant may have, no visibility to management of what has been accomplished and by whom, and - crucially - it provides no integration with IT systems that might play an important role in the process.

RADRunner is different. It is the only tool currently available that supports a modern, web-enabled version of Role Activity Theory, which was originally developed to describe the collaborative human process of software development. It provides more structure than groupware, but is less prescriptive and rigid than workflow and other machine-like process tools. In particular, RADRunner:

1. Captures and supports the particular goals and responsibilities of those involved in collaboration.
2. Integrates systems to meet the separate information needs of each participant - storing each participant's information resources in correct, coherent, and up-to-

date form, and making them available as and when required.

3. Provides for both structured and unstructured communication between participants.
4. Supports the processes in which the participants engage - with as much structure and/or flexibility as required.

RADRunner offers a way forward for organizations looking to provide IT support for collaborative processes. In particular, it may be the means for organizations who have already spent time and effort optimising their "machine processes" to advance to the next level of efficiency, for example as part of a Six Sigma and/or CMM initiative. Further work is underway to develop the principles underlying Playwright and RADRunner into a comprehensive methodology for the IT support of enterprise collaboration (see, for example, [25]).

5 Appendix: PML and Playwright

The original formal language for Role Activity Theory was *Process Modelling Language* (PML), a compiled (rather than interpreted) language developed in the IPSE 2.5 project during the late 1980's, and later incorporated into software systems such as *ProcessWise Integrator* and *ProcessWeb* ([4], [5], [6], [7], [8], [11]). The following description of PML is taken from [12]:

PML is an object based language. The basic building block[s] of an application are program objects called roles, and a means of communication between roles which are called interactions.

Some roles are programmed by the application writer. Such a role is an object that includes code (actions) and data (resources). It receives messages passed from other roles and from outside the system, processes them, and optionally send[s] back one or more replies. It encapsulates its data, and external access to its data is allowed only through special mechanisms intended for application modification or diagnostics. Some roles are standard and may not be modified by the application.

Interactions are uni-directional asynchronous communication channels. If one role wishes to communicate with another it requires two interactions, one to send and one to receive. Interactions are primarily a means of sending values not addresses, which helps to preserve the integrity of a role. It is possible to send roles and interactions through an interaction. In the case of a role a reference is sent, but a reference to a role is useful only for program control and diagnostic purposes. In the case of an interaction it is not the interaction as a whole which is sent, but one end of it, the giveport or takeport.

In this case integrity is maintained by removing the port from the role which is sending it.

PML is a class based language with single inheritance. There are three class hierarchies: Role, Entity and Action. A class may be based on the definition of another class in which case it inherits all its code. It may add to the class definition, or modify it, subject to certain constraints. With all three classes, inheritance is a means of reusing code. With entities subtyping is also supported.

A class defines a template for the creation of an instance of the object. In the case of a role, an instance is a program object which interacts with other roles. In the case of an entity an instance is an item of data. An Action class is the equivalent of a subroutine in other languages, and an instance is a particular call of an action. This instance has no life after the call has terminated.

Some example PML:

PersonDetails isa Entity with

parts

personName : String := 'name undefined'

personAge : Int ! default initial value will be nil

personMale : Bool := true

end with

Playwright, by contrast, is an interpreted XML language. It removes from classical RAT the basis in Petri Nets, which have a fixed connection structure, with an approach implementing the pi-calculus of Robin Milner [18].

This is done by including in the language full support for users as well as Roles, in addition to built-in task types for manipulating process types and instances. Hence you can use a Role to define, start, stop, monitor, and generally manage processes and their associated users.

Moreover, the interpreted nature of Playwright (as opposed to PML, a compiled language) means that changes to Role types or even instances are immediately effective - there is no need for a build-compile-deploy cycle to implement change.

An example Playwright fragment:

```
<nodeType>ActionNode</nodeType>
```

```
<name>Create Offer</name>
```

```
<preState>
```

```
<assertion>Bid Set AND Offer Not Made</assertion>
```

`<value>>false</value>`
`<enabled>>true</enabled>`
`<name>Create Offer</name>`
`</preState>`
`<enabled>>true</enabled>`

References

- [1] Holt A. W., Ramsey H. R., Grimes J. D., 1983, "Coordination system technology as the basis for a programming environment", Electrical Communication, Vol. 57(4).
- [2] Greenspan S.J., 1984, "Requirements Modelling: A Knowledge Representation Approach to Software Requirements Definition", University of Toronto Report CSRG-155
- [3] Ould M.A., Roberts C., 1986, "Modelling iteration in the software process", Proceedings of the Third International Software Process Workshop, Breckenridge, Colorado, USA, IEEE Computer Society Press.
- [4] Ould M.A., Roberts C., 1987, "Defining formal models of the software development process", Software Engineering Environments, Pearl Brereton (ed.), Ellis Horwood.
- [5] Snowdon R.A. , 1988, "A Brief Overview of the IPSE 2.5 Project", Ada User, Volume 9, No. 4.
- [6] Roberts C., 1988, "Describing and acting process models with PML", Proceedings of the Fourth International Software Process Workshop, Moretonhampstead, Devon, UK, IEEE Computer Society Press.
- [7] Roberts C., Jones A., 1990, "Dynamics of process models in PML", Proceedings of the Fifth International Software Process Workshop, Kennebunkport, Maine, USA, IEEE Computer Society Press.
- [8] Snowdon, R.A., Warboys, B.C., 1994, "An Introduction to Process-Centred Environments," Software Process Modelling and Technology, A. Finkelstein, J. Kramer and B. Nuseibeh (eds.), Research Studies Press, Taunton, England.
- [9] Ould M.A., Huckvale T., 1995, "Process Modeling - Who, What, and How: Role Activity Diagramming", Business Process Change: Concepts, Methods and Technologies, Idea Group Publishing
- [10] Ould M. A., 1995, "Business Processes: Modelling and analysis for re-engineering and improvement", Wiley, ISBN 0-471-95352-0.
- [11] Warboys B.C., et al, 1999, "Business Information Systems: a Process Approach", McGraw-Hill, ISBN 0-07-709464-6.
- [12] Department of Computer Science, University of Manchester, "CS637 Labwork Background (ProcessWeb)", <http://processweb.cs.man.ac.uk/doc/labbook4.pdf>
- [13] Arkin, A., 2002, "Business Process Modeling Language", Last Call Draft. <http://www.bpml.org/bpml-spec.esp>
- [14] Andrews, T, et al, 2003, "Business Process Execution Language for Web Services", version 1.1, <ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf>.
- [15] Workflow Management Coalition, 2001, "Workflow Process Definition Interface - XML Process Definition Language", version 0.03a, <http://xml.coverpages.org/XPDL20010522.pdf>.
- [16] OASIS ebXML Collaboration Protocol Profile and Agreement Technical Committee, 2002, "Collaboration-Protocol Profile and Agreement Specification", version 2.0, <http://www.oasis-open.org/committees/download.php/204/ebcpp-2.0.pdf>.
- [17] White, S., 2002, "Business Process Modeling Notation", version 0.9, <http://www.bpml.org/bpml-spec.esp>.
- [18] Milner, R., 1999, "Communicating and mobile systems: the pi calculus", Cambridge University Press, ISBN 052164320
- [19] Smith, H., Fingar, P., 2002, "Business Process Management - The Third Wave", Meghan-Kiffer Press, ISBN 0-929652-33-9.
- [20] Smith, H., Fingar, P., 2003, "Workflow is just a Pi process", <http://www.bpm3.com/picalculus>.
- [21] Hayden, F., et al, 2003, "The Networked Supply Chain", J. Ross Publishing, ISBN 1-932159-08-8
- [22] Harrison-Broninski, K., 2003, "From Requirements To Roll-out Immediately", <http://www.rolemodellers.com/download/unlicensed/FromRequirementsToRoll-outImmediately.pdf>.
- [23] Computer Sciences Corporation, "CSC e4 - Enabling the Process Managed Enterprise", http://uk.country.csc.com/en/kl/uploads/9_1.pdf.
- [24] Robert Shapiro, R., 2002, "A Comparison of XPDL, BPML and BPEL4WS"
- [25] Harrison-Broninski, K., Hayden, F., 2004, "Role-Based Transaction Management In Collaborative Systems", <http://www.rolemodellers.com/abstracts/Role-BasedTransactionManagementInCollaborativeSystems.pdf>