

Bridging the semantic gap between process documentation and process execution

Gregor Scheithauer*, Guido Wirtz

Distributed and Mobile Systems Group, University of Bamberg
Feldkirchenstraße 21, 96052 Bamberg, Germany
gregor.scheithauer@gmail.com, guido.wirtz@uni-bamberg.de

Candemir Toklu

Knowledge Management, Siemens Corporate Research, USA
candemir.toklu@siemens.com

Abstract

Process documentation and process execution are part of companies business information systems which describe the way how companies process information. Between these two levels of abstraction exists a gap that has been identified as the main drawback for business process implementation. This paper proposes a holistic methodology using BPM on top of SOA to overcome this gap by providing an architecture that bridges these different levels of abstraction by transition strategies, and a life cycle to support transitions.

Keywords: BPM, SOA, Process Modeling, BPMN, BPEL

1. Introduction

Process documentation and process execution are part of companies business information systems which describe the way how companies process information [6]. Two levels are differentiated within business information systems; The first level comprises business tasks which are associated with information relationships, whereas the second level is the task bearer level. The latter consists of a set of task bearers which are associated by communication systems. Between both levels exists an allocation relationship which represents the mapping of tasks and task bearers. This classification supports the understanding of the difference between process documentation and process execution. Process documentation is part of the task level. It represents the sequence of tasks which need to be performed in order to run a business. Also, it comprises the substantiation of business goals and strategies, and is carried out by people with business domain knowledge. The methodology used for process documentation is a visual depiction of processes [25]. The degree of formalization is either non-formal or semi-formal. The purpose of it is threefold: to reduce business process complexity, to ease communication between business partners, and to communicate business logic for the task bearer level. Process execution is associated with the task bearer level and represents software applications. The purpose of it is to support automated business processes. The solution is a formalized software code.

Mapping business tasks to software applications comprises several issues which need to be overcome in order to successfully map business requirements to software applications. The most prominent one, i.e., the loss of information during the mapping process [7, 19, 20, 23, 25], often results in a false translation from process models to implementation plans. The entirety of problems that occur in this context are often referred to as the *semantic gap*.

The aim of this paper is to offer an analysis of the semantic gap, and to examine the mapping between tasks and task bearers. Additionally, it addresses the issues described above by means of a holistic methodology, and offers a solution by narrowing the semantic gap.

Section two depicts work that is related to the holistic methodology. Section three presents a more detailed analysis of the semantic gap. Section four offers Business Process Management (BPM) [22] on top of Service-oriented Architectures (SOA) [25] as a holistic methodology to bridge the semantic gap. Section five summarizes the findings and presents future work to be done.

2. Related Work

The works of Decker, and Dehnert and van der Aalst also discuss interesting level concepts. Decker [4] differentiates between a business layer and an execution layer, and presents four strategies to map these. His work defines a process support layer, and patterns to support the mapping between the business layer and the execution layer.

Dehnert and van der Aalst [5] propose an approach to bridge the gap between business processes (EPC notation) and workflow specifications (Petri nets). Different levels of business process abstraction are addressed with different modeling techniques. The transition between different levels is implemented in a set of rules, which does not limit the designing facilities of the EPC notation.

Hofmeister's and Wirtz's [8] work relates to service integration. They present a pattern taxonomy to standardize the design of coupling systems and thus, to ease integration. To do this, they introduce a refined approach of Business Process Integration Oriented Application Integration (BPIOAI), which is based on message-based integration on a more ab-

*The first author is funded by means of the German Federal Ministry of Economy and Technology under the promotional reference 01MQ07012.

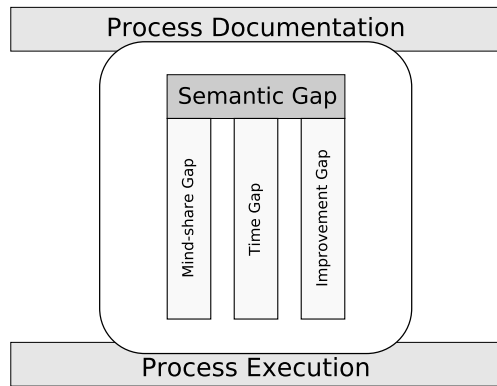


Figure 1. Different views on the semantic gap.

stract level. Regarding mapping and decomposition strategies, they propose an architectural framework to design composite applications on top of existing legacy applications. The framework includes service reuse, coupling, different layers of abstraction, and patterns.

Lastly, Model Driven Architecture is an approach by OMG to design software on the basis of formal models [11]. Its aim is to decrease the time to implement software applications. Platform independent formal models (PIM) are transformed into executable code.

3. Exploration of the Semantic Gap

The semantic gap is explained with the help of three different views to refer to the problems it implies; namely the mind share gap, the time gap, and the continuous improvement gap. Figure 1 summarizes these different views of the semantic gap between process documentation and process execution.

The *mind share gap* refers to the difficulty to translate business requirements directly into supporting applications. Business owners and software developers talk different languages, perceive problems and solutions differently, apply different methodologies, and use different approaches to communicate and understand business processes and requirements. These different contexts make it difficult to transform business requirements into executing software applications [20, 23].

The *time gap* refers to the time delay between the setting of business requirements and the development of the software applications. Software applications which support business logic are complex, involve a team of developers, need to be reliable, and need to be set up for a whole company. Hence, a great amount of time elapses between the moment of ordering the system and the moment of completion. In the worst case this could mean that business requirements change during software application development and consequently, the application becomes obsolete [7, 21].

The *continuous improvement gap* deals with the ability to refine business processes and the underlying supporting software applications. According to Woodley and Gagnon [25], simple steps in a business process do not change often. What does change often, are sequences and the integration of these into different business contexts. Right now, business require-

ments are hard-coded into software applications and cannot be changed easily. The association between original business processes, and applications supporting the process, is not formally captured and cannot be repeated or managed when processes need to be changed in order to adapt to new market needs. Moreover, as it is crucial for companies to accommodate their business processes according to their visions and business goals to match market requirements, it is not desirable anymore to hard-code business logic into one application but to store the business logic separately from implementation code. However, in a fast changing business environment, business requirements need to be adapted frequently [7, 21].

4. Holistic Methodology

This section presents a solution to bridge the *semantic gap*. The holistic methodology comprises (1) a level concept to clearly separate the task level and the task bearer level, (2) a life cycle to drive the whole methodology, and (3) transition strategies, which provide relations between levels. Every level serves as a classification for BPM goals, views on the semantic gap, the BPM life cycle, BPM stakeholders, design notations, execution languages, and technology. The level concept will be the basis for the transition strategies. Transition strategies are the bridge between process documentation and process execution.

The emerging BPM [22] technologies are providing an opportunity to manage the life cycle of the business processes for a company and therefore, effective deployment and execution of information technology [21] in order to support the bridging of the *semantic gap*. The BPM definitions from zur Muehlen and Ho [26], Gartner [10], van der Aalst et al. [22], and the BPM Standard Group¹ were considered for the derivation goals for BPM. Table 1 summarizes the findings about the essential BPM goals. Thus, BPM is a methodology to discover and document processes as well as to integrate software applications to support business processes in an automatic way, and it offers a life cycle to improve and adapt business processes according to changing business environments.

SOA supports BPM by offering a platform that supports business processes and integrates heterogeneous business applications and business partners. Business processes consume and leverage such SOA services, tying them together to solve business challenges [25]. BPM on top of SOA as a methodology provides a solution to narrow the semantic gap by solving the mapping problems and thus, to diminish the consequences. The independence between business processes and services helps to separate the business model and the technical implementation. The velocity of the implementation matches the speed of the quickly changing business requirements. Processes need to be independent from a specific platform. Otherwise, the logic is hard-coded into software applications and thus, expensive to change. Woodley and Gagnon [25] claim

¹BPM SG, <http://www.bpmstandardsgroup.org/resources.asp>

Table 1. BPM definition comparison.

	zur Muehlen and Ho	Gartner Group	BPM SG	van der Aalst et al.
Business driven	X	X	X	
Flexibility	X	X		X
Integration	X			
Improvement		X	X	X
Automation		X	X	X

that both BPM and SOA are IT concepts. This is only partly true. IT is only one part of the BPM methodology since so many more aspects need to be considered, such as management issues, business strategies, and people. However, SOA serves as an integration platform for business services that are loosely coupled and easy reused [9, 12, 15, 20].

4.1. The Level Concept

This subsection introduces a three level concept: the business level, which represents business process documentation and improvement, and the integration level, which represents a SOA implementation. It orchestrates services that support business processes [15]. The third level is the execution level that represents business applications.

The *business level* addresses the process documentation. Business processes, business tasks, business objects, and business partners are documented with the aid of business process diagrams designed with process notations such as Business Process Modeling Notation (BPMN) [24] and Event-driven Process Chains (EPC) [17]. Business processes are fundamentally very abstract. They consist of a flow of business tasks connected by a control flow. Business objects refer to information that is processed by business tasks. The interaction between two companies within a business process describes business partners. Business logic is refined by process decomposition. This makes it possible for business analysts to describe the requirements for supporting software applications in detail and in a language that is understood by business analysts. Furthermore, the flexibility of process design notations allows to express business logic without limitations. This narrows the mind share gap. Lastly, ongoing improvement allows the continuous adaption of business processes to a changing business environment, which leads to a narrowing of the improvement gap. Regarding the life cycle management process design, process monitoring, and process improvement occur on this level by business analysts, process owners, and business owners. This level is neither technical nor platform specific. The flexibility to change business processes is very high.

The *integration level* serves two purposes: firstly, it is the target platform for formal and executable processes; secondly, it provides the means for a common uniform representation for services. Business Process Execution Language (BPEL) [1] allows the implementation of complex business processes on the basis of existing web services. The BPEL code represents the sequence flow of business processes registered on the business level. The service functionality is described

with a specification such as the Web Service Description Language (WSDL) [3]. This specification also includes message type definitions which are processed by the service. A service represents an atomic business function. Otherwise, too much interaction is needed between services and partners. An integration-oriented architecture allows to integrate heterogeneous business applications onto a homogeneous level, which makes it easy to access and combine business applications to support business processes completely. Furthermore, the decoupling of services eases the change of business processes. In that case business logic may be changed on the business level. This change does not require an adaptation of a hard-wired business application. In fact, it is adequate to rearrange the supporting service using tools without the need to change application code, to recompile the code, and to redeploy the code with all the difficulties involved in application change management. This fact addresses the time gap and the continuous improvement gap. The integration level comprises the life cycle steps process configuration, service integration, and service development. The system analyst is responsible for process configuration, and service integration, whereas the software developer is responsible for developing the services. This level encounters middleware solutions, such as EAI, process configuration tools, and tools to deploy configured processes. The integration level is the soft glue between flexible business processes and hard-coded applications. Though the integration level is technical, it is still not platform specific. It is possible to take process configurations and deploy them in a different technical environment that supports open standards.

The *execution level* addresses business applications, legacy code applications, process execution engines, and other middleware. It forms the technical basis for automated business processes. Many business functions are hidden within business applications, which makes it difficult to use or even reuse them in loosely coupled business processes. The execution level comprises the life cycle steps service development, process deployment, and process execution. The software developer is responsible for service development and is involved in the process deployment step. This level encounters development and middleware tools. Thus, this level is technical and platform specific.

4.2. Life Cycle

Using the level concept, a life cycle to transform business process diagrams into executable processes will be introduced. Table 2 summarizes the findings of the life cycle concepts of Smith and Fingar [21], Netjes et al. [14], and the BPM Group.

Table 2. BPM life cycle comparison.

	Smith and Fingar	Netjes et al.	BPMG
Process Discovery	X	X	X
Process Design	X	X	X
Process Configuration		X	
Service Integration			X
Service Development			X
Process Deployment	X		
Process Execution	X	X	
Process Monitoring	X	X	X
Process Improvement	X		X

A comparison of the three life cycle concepts, resulted in a nine step life cycle for managing business processes.

Process discovery refers to the detection of business goals and strategies in order to conduct a business. A way to formalize goals and strategies are either ontologies or a business model [16].

Process design refers to the transformation of goals and strategies into a process diagram on the business level. In order to perform this step, two intermediate steps are required: first, business analysts will need a business process design notation. In this context, this might either be the BPMN or the EPC notation. Subsequent, business analysts will use their experience, design paradigms, and the purpose of the documentation for transforming the informal goals, strategies, and rules into a process documentation. The design paradigm is process-oriented. Business analysts decompose tasks and objects to substantiate business processes. Second, they need to simulate the process in the diagram to verify the semantical behavior of the process.

Process configuration refers to the transformation of process documentation into a platform independent process configuration. It implies the mapping strategy on the integration level. It is intended to transfer the process documentation as complete as possible, not to change any business logic, and to reuse existing implementations. To achieve this step, system analysts need to perform four steps: First, the process documentation is transformed into a technical representation. At this point, a switch in notation is not necessary. The configuration tool offers additional constructs to enrich the process documentation with technical details. Hence, the flow of business tasks (service composition) can be derived from the business process diagram. Second, web services are mapped to atomic business tasks. The integration is done by using middleware technology, such as webservices. Third, message descriptions need to be applied to web services. They also need to be maintained and integrated by the system analyst. Fourth, system analysts need to validate the work and build the process configuration. In case no service for an atomic business task is available, two possibilities are conceivable: consultation of a software developer, who develops an application function to address the business tasks requirements, or to integrate a service from a service provider.

Service integration follows either the development step, or

the configuration step. It supports the integration strategy. Integration is twofold. Both, developed services, and services from business partners need to be integrated before they can be used for process configurations.

The development of an application function that is wrapped as a service and fulfills the requirement of an atomic business task is called *Service development*. The objective is to develop a service which is not bound to a specific business process resulting in a self contained service which is easy to reuse. Since business analysts break business processes down into loosely coupled business tasks and decompose business tasks (treated as business processes) recursively into atomic business tasks, the whole complexity and semantic is broken down into seizable and easy to communicate requirements, which are understood by software developers. Software developers grasp the requirements and build an application function using programming languages such as Java, or .NET and wrap this function as a reusable service.

Process deployment refers to the final transformation of the process configuration onto a platform. It supports the transformation strategy. However, the deployment should not have an influence on other running processes on that platform. The executable process configuration needs to be transferred to the execution engine. The configuration of other middleware technologies, which are involved in the process, as well as security policies, is optional.

Process execution alludes to the state where configured business processes are executed to support business processes. The procedure to execute a process differs in terms of how processes are triggered and whether the service is known in the first hand. First, process consumers need to know about the service. Second, processes need to be triggered. Process might be triggered by an event, other services or by human interaction through a process portal. Third, running processes are referred to as process instances. Process instances save the state of a process, as well as message and variable values.

The tracking of the process performance and to display it in a human understandable format is referred to as *Process monitoring*. The formal goals are to highlight bottlenecks and offer suggestions for action in order to improve process performances. Often, a dashboard is used as a paradigm. Business owners should be in the position to view the performance of process instances on a consolidated level. It should be al-

lowed to break the level down to a single instance which may be a source of failure or a bottleneck. Business owners are then allowed to take action, e.g. improve the process.

Process improvement points to the adaption of processes due to a changing situation or environment. Objectives are to improve execution time, execution cost (total cost of ownership), and process efficiency. Business analysts and business owners gather new information and feedback from process monitoring, process users and business owners. They change the process diagram according to the new requirements. The new process diagram needs to be validated. Moreover, depending processes need to be checked whether they are affected by the current change of the original process. Adapted processes are configured and deployed for execution.

4.3. Transition Strategies

Transition strategies address the transition between the business level and the integration level, and the integration level and the execution level. The mapping between the execution level and the integration level will be carried out once for every service on the basis of standards. The mapping between the business level and the integration level occurs every time a new business process is introduced or a business process is adapted, and thus, new configured and deployed.

The *decomposition strategy* starts with documented business processes and business objects using a process notation, such as BPMN, according to business strategies, goals, and rules, provided by business owners and process owners. These documentation may serve as a communication basis between business owners and business analysts. However, to communicate business logic to process participants on a more operational level, it is necessary to concretize the business logic. This is done by business analysts in connection with business owners. They *decompose* abstract business tasks. This is possible if business tasks are treated as processes themselves. The process notations BPMN and EPC support the decomposition strategy. Decomposition is an iterative procedure. In case a business task cannot be decomposed anymore without losing business relevance, the decomposition stops. Tasks which cannot be decomposed anymore are referred to as atomic business tasks. Though decomposing business tasks reduces the complexity of business logic, the reduction in complexity has its limits, if business language is enriched with too many details. Furthermore, this hierarchical adjustment of business tasks allows business analysts to change business processes easily. The decomposition strategy narrows the mind share gap and the improvement gap.

The *mapping strategy* urges business analysts to create self-contained atomic business tasks for three reasons. Self contained business tasks improve the reusability of these business tasks, since they are applicable in more than only one process. Furthermore, self-contained business tasks are *mapped* more easily to stateless web services [25]. The semantics of self-contained business tasks is communicated to

system analysts more easily, since no context knowledge is necessary. Thus, atomic business tasks are the ones to be mapped to services on the integration level, and atomic business objects are the ones to be mapped to message types. In conclusion, services must be available which correspond to atomic business tasks. According to Natis [13], the reason for coarse-grained services is that the message interaction between services does not need to be chatty. System analysts take over at this point. They are responsible to map atomic business tasks to business services. They need to search for an appropriate service, which might be a difficult task, in case the service repository is huge, and in case the semantic of the atomic business task is not well understood by the system analyst. In both cases, system analysts may consult business analysts for support, or system analysts are supported by clever tools. The mapping strategy addresses the time gap and the mind share gap.

The *integration strategy* is carried out by system analysts. System analysts are responsible for the service repository. They integrate services and message types from within the company and from external service providers. Thus, it is possible to use powerful software applications, in a standardized fashion. Business processes do not stop at application borders, they cross application functionality as well as department responsibilities. However, it is not intended to *integrate* services on the basis of availability. Since BPM should be business-driven, requirements for atomic business tasks and atomic business objects should be the trigger to develop or integrate a service. Thus, system analysts in connection with business analysts define requirements for services which need either to be developed by software developers on the basis of existing software applications, or to be searched in service repositories. The integration strategy supports the integration goal of the BPM methodology, and addresses the time gap.

The *transformation strategy* refers to the procedure of transferring the atomic business task orchestration [18] into executable code. Business analysts decompose abstract business processes into an orchestration of atomic business tasks. System analysts map these atomic business tasks to corresponding business services. The configured process is now ready to be transformed into executable code. All information required is provided by the mapping of atomic business tasks to web services. Executable code refers to an execution language which may be executed by execution engines, like, e.g., BPML [21], BPEL [1], or a vendor specific execution language. The transformation procedure is an automatic step, thus it is easy executed and does not limit the improvement of business processes. Transformation supports the automation and improvement goal, and addresses the time gap.

5. Conclusion and Future Work

The aim of this paper was to bridge the semantic gap between process documentation and process execution. Problems originating from the semantic gap were identified and classified into the three different views. BPM on top of SOA was in-

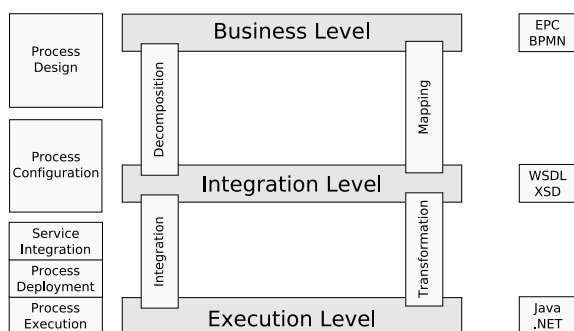


Figure 2. Architecture: Levels & Strategies.

troduced as a platform for a solution. The methodology comprises three levels, four transition strategies, and a life cycle. Figure 2 illustrates how the four transition strategies link levels, as well as how the life cycle corresponds to the levels and the transition strategies. In conclusion, the combination of the presented holistic methodology is capable to bridge the semantic gap.

Future work needs to address the strategic level and the transition between the strategic level and the business level. The strategic level is on top of the business level and covers business strategies and business goals, which are handled by business owners and process owners. Furthermore, existing process design notations need to be improved regarding the semantic of elements, and the methodologies how to use the notations to express processes. The EU-aided IP-Super project² addresses this question. Standards must be available to easily exchange process diagrams between stakeholders and tools. Process execution languages need to include more concepts to match business requirements. Patterns and standards for the transformation of process design notations into process execution languages must be improved. Moreover, existing tools and technologies must collaborate better to match requirements for BPMS. Finally, business-to-business integration must be raised to another level. Approaches, and tools must come available for the upcoming concept of service ecosystems, which Barros and Dumar depict in [2].

References

- [1] A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Golland, A. Guzar, N. Kartha, and C. K. Liu. Specification: Business Process Execution Language for Web Services vers. 2.0. Tech. rep. 2.0, OASIS, Jan. 2007.
- [2] A. P. Barros and M. Dumas. The rise of web service ecosystems. *IT Professional*, 8(5):31–37, 2006.
- [3] D. Booth and C. K. Liu. Specification: Web Services Description Language (WSDL) Version 2.0 Part 0: Primer. Technical report, W3C, March 2006.
- [4] G. Decker. Bridging the Gap between Business Processes and existing IT Functionality. In *Proc. 11th Int. WS on Design of Service-Oriented Applications (WDSOA '05)*, 2005.
- [5] J. Dehnert and W. Aalst. Bridging the Gap between Business Models and Workflow Specifications. *International Journal of Cooperative Information Systems*, 13(3):289–332, 2004.

- [6] O. Ferstl and E. Sinz. Modeling of Business Systems Using SOM. *Handbook on Architectures of Information Systems*, 1:347–368, 2005.
- [7] M. Hammer. Reengineering work: Don't automate, obliterate. Technical report, Harvard Business Review, July 1990.
- [8] H. Hofmeister and G. Wirtz. A multi-layered framework for pattern-aided composite application design. In *Proc. of the 11th World Multi-Conference on Systemics, Cybernetics and Informatics*, July 2007.
- [9] F. Leymann, D. Roller, and M.-T. Schmidt. Web Services and Business Process Management. *IBM Systems Journal*, 41:198–211, 2002.
- [10] M. J. Melenovsky, J. Sinor, J. B. Hill, and D. W. McCoy. Business Process Management: Preparing for the process-managed organization, June 2005.
- [11] J. Miller and J. Mukerji. *MDA Guide Version 1.0.1*. Object Management Group, Framingham, MA, June 2003.
- [12] Y. Natis. Service-Oriented Architecture Scenario. Gartner Research, ID Number: AV-19-6751, April 2003.
- [13] Y. V. Natis. Service-Oriented Architecture (SOA) Ushers in the Next Era in Business Software Engineering. *Business Integration Journal*, May 2004:23–25, May 2004.
- [14] M. Netjes, H. Reijers, and W. v. d. Aalst. Supporting the BPM life-cycle with Filenet. In 18th Int. Conf. on Advanced Information Systems Engineering (CAISE'06), ed., *Proc. of the EMMSAD Workshop*. Namur University Press, 2006.
- [15] J. Noel. BPM and SOA: Better Together. IBM Website, White Paper, 2005.
- [16] A. Osterwalder, C. Parent, and Y. Pigneur, editors. *Setting up an Ontology of Business Models*. Faculty of CS and Information Technology, Riga Techn. Univ., Riga, Latvia, 2004.
- [17] A.-W. Scheer and M. Nuettgens. Architecture and Reference Models for Business Process Management. *Lecture Notes in Computer Science*, 1806 / 2000:376–389, 2000.
- [18] A. Schoenberger and G. Wirtz. Using Webservice Choreography and Orchestration Perspectives to Model and Evaluate B2B Interactions. In *Proc. of SERP 2006*, volume 2006, pages 1–7. CSREA Press 2006, June 2006.
- [19] H. Smith. Business Process Management—the third wave: Business Process Modelling Language (BPML) and its Pi-calculus foundations. *Information & Software Technology*, 45(15):1065–1069, 2003.
- [20] H. Smith. The Emergence of Business Process Management. *Information & Software Technology*, 45(15):1065–1069, December 2003.
- [21] H. Smith and P. Fingar. *Business Process Management, the third wave*. Meghan-Kiffer Press, 1th edition, January 2003.
- [22] W. M. P. van der Aalst, A. H. M. ter Hofstede, and M. Weske. Business Process Management: A Survey. In W. M. P. van der Aalst, A. H. M. ter Hofstede, and M. Weske, editors, *Business Process Management*, volume 2678 of *Lecture Notes in Computer Science*, pages 1–12. Springer, June 2003.
- [23] L. Verner. BPM the Promise and the Challenge. *ACM Queue: Tomorrow's Computing Today*, 2(1):82–91, March 2004.
- [24] S. A. White. Specification: Business Process Modeling Notation Specification, February 2006.
- [25] T. Woodley and S. Gagnon. BPM and SOA: Synergies and Challenges. In A. H. H. Ngu, M. Kitsuregawa, E. J. Neuhold, J.-Y. Chung, and Q. Z. Sheng, eds., *WISE*, Vol. 3806 of *Lecture Notes in Computer Science*, pages 679–688. Springer, 2005.
- [26] M. zur Muehlen and D. T.-Y. Ho. Risk Management in the BPM lifecycle. In C. Bussler and A. Haller, eds., *Business Process Management Workshops*, vol. 3812, pg. 454–466, 2005.

²IP-Super, <http://www.ip-super.org/>, last accessed 2008-02-29