A Framework for Querying in Business Process Modelling

Ivan Markovic[1], Alessandro Costa Pereira[1], Nenad Stojanovic[2]

[1]SAP Research, Karlsruhe, Germany
ivan.markovic@sap.com
alessandro.costa.pereira@sap.com

[2]FZI Karlsruhe, Germany
nenad.stojanovic@fzi.de

Abstract: In order to respond quickly to changing market requirements, a business organisation needs to increase the level of agility in all phases of the business process engineering chain. Business process (BP) modelling is the first and most important phase in this chain. Designing a new and redesigning an existing process model is a highly complex, time consuming and error prone task. In this work, we contribute to BP modelling by designing and implementing a framework for querying in business process modelling which i) supports decision making, ii) facilitates reuse of modelling artefacts and iii) helps ensuring compliance of models to relevant regulations.

1 Introduction

In the modern world, businesses constantly strive to reinvent and differentiate themselves under continuous pressures of regulatory and technological change, as well as the increasing time to market requirements. One of the main obstacles for the changes to be agile is the lack of support when incorporating new business requirements into existing information systems as priorities and perspectives change.

Business process (BP) modelling is the first and most important phase in the business process engineering chain. BP models are created by business analysts with an objective to capture business requirements, enable a better understanding of business processes, facilitate communication between business analysts and IT experts, identify process improvement options and serve as a basis for derivation of executable business processes. Designing a new process model is a highly complex, time consuming and error prone task. This is because BP modelling involves several sources of information, models are dynamic and frequently redesigned to adapt to changes, and BP models are often shared by several departments within a company or even between different companies.

In order to simplify BP modelling, models must be highly reusable, favoring process flexibility and minimizing designs made from scratch. Reusing implies the need for querying the process repository in order to find suitable previous work that can be the base for a new design. This can be done only by an expressive and machine-readable description of relevant aspects of a BP model that will help to retrieve the most relevant parts of a previous work (model).

Therefore, in order to enable expressive querying of BP models, there is a need for a comprehensive formal process model description capturing all relevant dimensions (perspectives) of a process. Following [CKO92] and [JB96], we consider the functional, behavioural, organizational and informational perspective relevant to adequately organize information about a process. Based on these requirements, in our previous work [MP07] we have proposed a formal model for describing business processes which integrates all aforementioned perspectives, as depicted in Figure 1.

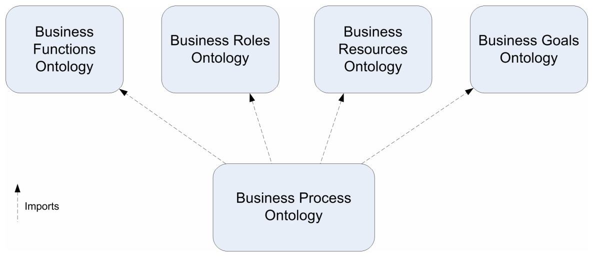

Business Functions Ontology
Business Roles Ontology
Business Resources Ontology
Business Goals Ontology
Imports
Business Process Ontology

Figure 1: Ontology framework: Perspectives for describing a business process

In the following we are briefly describing these perspectives.

For describing the behavioural (dynamic) perspective of a process model we use a process algebra, the π-calculus. By using the π-calculus for representing the process behaviour, we are also able to integrate existing tools and techniques for verification and simulation of processes [Pu06, AB06] in our framework. The dynamic perspective of a process model stands for process control- and dataflow, and it we model it using the ontologized π-calculus, denoted by Business Process Ontology in Figure 1. For more details on this ontology, we refer the reader to [MP07].

For representing the functional, organizational and informational perspective we have proposed a set of ontologies, imported by Business Process Ontology, as shown in Figure 1. Business Functions Ontology provides a structural breakdown of the organization’s business functions. Concepts from this ontology classify process models by their functionality, independent of the business domain. Business Roles Ontology includes concepts representing roles in the organization, e.g. Manager, Engineer, Clerk, Secretary, etc. Business Resources Ontology describes the resources (documents, systems, machines) which are required to operate the activities in processes. Business Goals Ontology models a hierarchy of organization business goals (milestones, objectives) according to which the processes in the organization are designed. Business goals are modeled in such a way that they conflict if they can not be satisfied simultaneously. Moreover, goals can influence positively or negatively other goals [YM94]. Note that we refer to these perspectives as static view of a process in the rest of the text.

Having captured all relevant perspectives of a process in our model, we expose the complete process description to advanced querying and reasoning. In this paper we describe the framework for querying business process models and explore its formal nature in details.

The paper is structured as follows. In Section 2, we present three sample usage scenarios for our framework. Section 3 describes the framework in general. In Section 4, we show how the querying process operates inside our framework. Section 5 discusses related work. We conclude and give an outlook on future research in Section 6.

2 Motivating Examples

To illustrate the need for a querying framework in business process modelling, we discuss several example scenarios of the framework usage in this section.

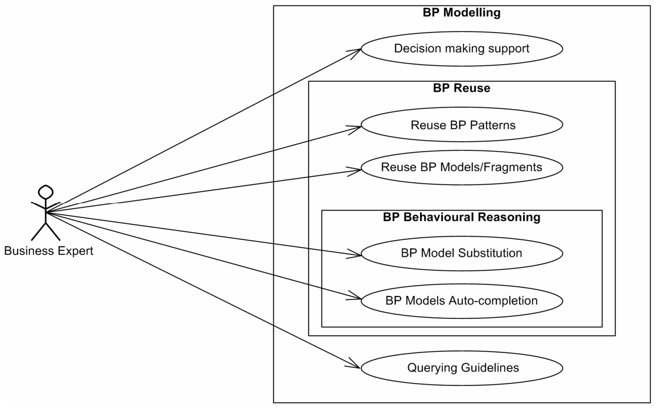


Figure 2: Usage scenarios for the querying framework

Scenario 1: Decision making support

The key challenge in decision making is having access to all relevant information which is to be assessed in a particular situation. Such information is scattered in organization processes and has to be manually collected from diverse sources for each individual case. To facilitate this task, we enable the business expert to quickly and expressively query the process artefact[1] repository of an organization (cf. Figure 2, top). Some example queries for this scenario include: "*Give me all processes in the fulfillment area*", "*Which processes use system x*?", "*What resources are needed for running process y*?", "*List all processes with conflicting goals*." [He05].

[1] Note that we refer to process patterns, models, fragments and modeling guidelines as process artefacts

Scenario 2: Reuse of process artefacts

This scenario describes how the business expert can query the process artefact repository for reuse of process patterns, models and fragments in process design (cf. Figure 2, center). Since process modelling is a complex activity, reuse of existing models and model components makes sense in all stages of modelling. For instance, when designing a new process the business expert can first query for existing business process patterns, generic high level process designs emphasizing business goals [Ma07], in search for the best modelling practices in the given domain. An example query for business patterns can be: "*Give me all business patterns related to Fulfillment Business Function where Business Goals involved are profileObtained and serviceActivated*". The business expert can also query in the same way for existing models or process fragments - self-contained, coherent building blocks of a process model with a clear business meaning. In case that there are existing process models or fragments that are similar to the desired end design, the business expert can use them in his design in order to achieve a higher degree of reuse, compared to reuse of patterns. Moreover, if the user wants to substitute an existing process fragment based on redesign goals or auto-complete an underspecified model, he can make graphical queries by selecting the desired process part for the substitution or auto-completion in the modelling tool. For this purpose we use properties of bisimulation theory for the π-calculus [Sa96](cf. Figure 2, center).

Scenario 3: Querying modelling guidelines

This scenario covers querying for business guidelines – concrete policies defined according to the company strategy, which apply orthogonally to all processes of an organization (cf. Figure 2, bottom). Queries involved in this scenario retrieve all modelling guidelines (both mandatory and conditional) which match context annotations of the model being checked. This reduces the manual effort of creating an inventory of such guidelines for any given model. For checking which guidelines are relevant in a digital content provisioning process, the example query can be: "*Give me all modelling guidelines for Digital Asset Management Business Function where clients are minors and Business Goal associated belongs to Fulfillment*".

For more details on different types of queries that business expert can perform in process modelling, we refer the reader to [Ma07].

3 Querying Framework

In order to realize the aforementioned scenarios, we have derived a set of requirements described in [MP07]. Based on the requirements, we developed a framework for querying in business process modeling. This section presents the components of our querying framework, discusses their functionalities, relation and communication with other components.

3.1 Framework architecture

In Figure 3 we illustrate the framework components and their interactions using a block diagram in FMC notation[2]. The framework consists of existing components that have been reused, new components that have been implemented, integration techniques and interfaces for external access. The basis for communication between components is the ontological process description based on the ontology framework in Figure 1.

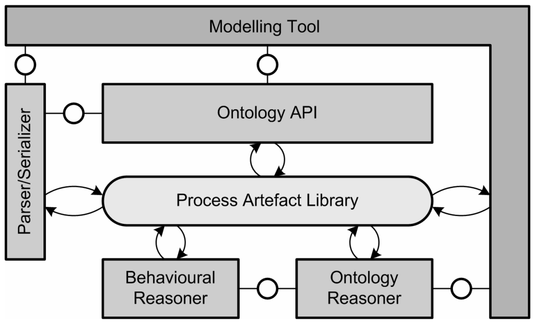


Figure 3: Framework architecture

A business *Modelling tool* has the function to provide an environment for modelling business processes. It is a component which is used by the business expert to interact with our framework. In particular, the user interacts through a user-friendly query interface which will be detailed later in this work. We extended the SAP Research modelling tool "Maestro for BPMN" for integration with our querying framework.

The *Process artefact library* has the function to provide persistent storage for process patterns, models, fragments and guidelines. For this purpose we used the Ontology Representation and Data Integration (ORDI)[3] framework, a middleware component designed to be able to load various ontology languages and allow enterprise data integration via RDF-like data model.

The *Ontology reasoner* performs ontological reasoning for obtaining the query results. Since behavioural reasoning is computationally more expensive, query answering by the ontology reasoner is performed first and serves as a filtering step to obtain a subset of process descriptions for later behavioural conformance checking (cf. Figure 3). Since our ontologies are represented in the WSML[4] language, we use WSML2Reasoner[5] framework on top of IRIS[6] back-end reasoner to perform ontological reasoning.

[2] Fundamental Modeling Concepts, http://www.fmc-modeling.org/
[3] http://ordi.sourceforge.net/
[4] Web Service Modeling Language (WSML) – Final Draft, http://www.wsmo.org/TR/d16/d16.1/.
[5] http://tools.deri.org/wsml2reasoner/
[6] http://iris-reasoner.org/

The *Behavioural reasoner* is a component that performs π-calculus reasoning, i.e. it serves as a query answering mechanism for graphical queries used in process substitution and auto-completion scenarios. At the moment this component implements the strict congruence algorithm, based on the definition of bisimulation from [MPW89]. In the future, we plan to embed Mobility Workbench (MWB) [VM94] as a subsystem in our framework implementing bisimulation for the π-calculus. This will allow us to investigate more advanced notions of process equivalence.

The *Ontology API* provides methods for creating and manipulating the ontology object model. This API is used by the modelling tool for creating an in-memory representation of the modelled process, based on the ontology framework in Figure 1. For our purposes we utilized WSMO4J, a reference implementation for WSMO[7] and WSML specifications.

The *Parser/Serializer* component has been implemented to provide the round trip transformation between WSML ontological representation and plain π-calculus representation describing the behavioural perspective of a process.

In the rest of this section we give more details about the user interface component, embedded in the modelling tool.

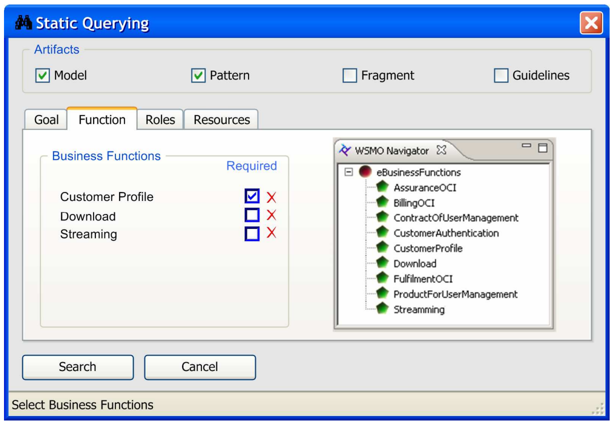
3.2 User-friendly query interface

The intended users of our framework are business experts. Therefore, they need to be provided with an intuitive and user-friendly query interface to be able to specify their queries in an easy way, not much different from using the applications they are used to. The complexity of ontologies and reasoning needs to be hidden from the user. In the following, we describe how we designed the query interface based on this requirement.

The query input dialog for performing the queries on the static view of a process (static queries) is presented in Figure 4. The user can navigate through tabs for selecting business annotations (business goals, functions, roles and resources) of the processes he wants to retrieve. Note that these characteristics correspond to the perspectives depicted in Figure 1. In the right box, the ontology navigator provides available ontology concepts which the user can browse.

Desired business annotations can be dragged from the ontology navigator to the left box (Business Functions panel) and marked as required or optional for querying. This is important for achieving flexibility in querying, since the concepts marked as optional can be omitted when constructing the query for retrieving more results (cf. Section 4.3). The user can also specify the type of artefacts he is looking for (model, pattern, fragment, guideline) in the upper panel.

[7] Web Service Modeling Ontology (WSMO) – Final Draft, http://www.wsmo.org/TR/d2



Artifacts

[✓ Model] [✓ Pattern] Fragment Guidelines

Business Functions

	Required
Customer Profile	✓
Download	☐
Streaming	☐

eBusinessFunctions
- AssuranceOCI
- BillingOCI
- ContractOfUserManagement
- CustomerAuthentication
- CustomerProfile
- Download
- FulfilmentOCI
- ProductForUserManagement
- Streamming

Select Business Functions

Figure 4: Static querying interface

In contrast to performing static queries, querying for desired behaviour does not need a new dialog. Querying is performed by selecting a process part directly in the modelling tool, as shown in Figure 5 (shaded area). We consider this way of graphical querying to be intuitive to the user. The behavioural description of the selected part is obtained automatically using the Parser/Serializer component and used as an input for behavioural reasoning (substitution, auto-completion, deadlock/liveness verification).

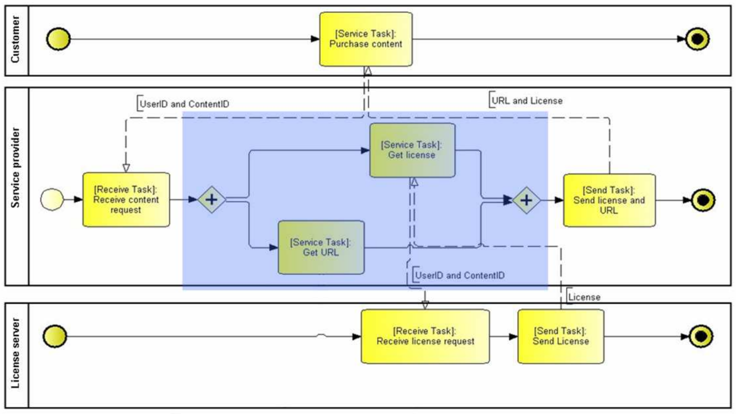

Customer
[Service Task]: Purchase content
Service provider
UserID and ContentID
URL and License
[Receive Task]: Receive content request
[Service Task]: Get license
[Service Task]: Get URL
[Send Task]: Send license and URL
UserID and ContentID
License
License server
[Receive Task]: Receive license request
[Send Task]: Send License

Figure 5: Dynamic querying interface

4 Querying Process

In this section, we discuss how the querying process operates inside our framework. We first discuss the formal query language which has been defined for coupling static and dynamic process characteristics. Further, we shortly describe the mechanism for processing queries. Third, we explain how we have introduced flexibility in querying for providing better query results.

4.1 Formal query specification

The query must support the description of static and dynamic behaviour. WSML Logical Expression (LE) is used for specifying queries on static properties of a process. The query should reference instances of business annotations, which are materialized by instances of the relations described in [MP07]. An example is given in the following:

bpo#hasBusinessFunction(?x,Annot1) **and** bpo#hasBusinessRole(?x,Annot2) **and** bpo#hasBusinessResource(?x,Annot3) **and** bpo#hasBusinessGoal(?x,Annot4)

The dynamic behaviour of a wanted process is described as a process definition, i.e., it uses the ontology framework given in Figure 1 for describing a process, its connections, and the annotations of tasks as relation instances. Defining that the behavioural query has the same structure as the process definitions avoids the mismatch problem that would appear from having a different query language for behavioural querying. Therefore, here we use the ontologized π-calculus to describe the user request (process query). This query is checked against processes stored in the repository using congruence and bisimulation properties.

The query specification must be defined to use the same language used by processes description. This enables the reuse of the ontological process model for representing the user queries. For meeting our requirements, the query specification should be in a format of a template with placeholders. In addition, it should encapsulate both static and dynamic attributes in a unified language.

The use of a BPO ontology instance is the solution to this: the query template corresponds to a pre-defined ontology structure with namespace definition and element descriptions. Table 1 shows the template contents:

Namespace	http://www.ip-super.org/ontologies/BPO/extension/query
dc:type	Either “substitution” or “autocompletion”
Axiom	ID:http://www.ip-super.org/ontologies/BPO/extension/query#static definedBy: bpo#hasBusinessResource(?x, _PLACEHOLDER_) ‘OR’ bpo#hasBusinessFunction(?x, _PLACEHOLDER_) ‘OR’ bpo#hasBusinessGoal(?x, _PLACEHOLDER_) ‘OR’ bpo#hasBusinessRole(?x, _PLACEHOLDER_)
Process	ID:http://www.ip-super.org/ontologies/BPO/extension/query#dynamic A BPO process model description.

Table 1: Query definition using BPO ontology instance

The ontology contains an axiom (containing the static query), and a process definition (containing the process behaviour). The non-functional property dc:type indicates the type of behavioural query being performed.

The ontology submitted as query contains all necessary information for specifying the query request. The approach enables performing ontological reasoning on the static part and π-reasoning on the dynamic part of process description. This approach is also scalable, since adding new concepts or adding new information to the query will not imply changes to the established query definition.

4.2 Query mechanism

In order to reduce the level of complexity, we have divided this task into two subtasks – static and dynamic (behavioural) querying. The querying mechanism operates on the Business Process Ontology (BPO), presented in Figure 1.

The first subtask investigates simple (static) querying, where the user can specify constraints related to the static view of a process. Here we use WSML logical expressions as a query language and ontological reasoning for query answering. The second subtask investigates graphical (behavioural) querying, where the user can specify requirements on dynamic perspective of a process description. This corresponds to autocompletion and substitution scenarios where algorithms from bisimulation theory are used for comparing the processes. In case that the user request contains constraints on both static and dynamic perspective of a process, static querying is performed first since dynamic querying is more computationally expensive. For more details on the framework querying mechanism, we refer the reader to [Ma07].

4.3 Achieving flexibility in querying

Depending on the user query, there can be too many or too few results coming from the repository. The user should be able to i) specify further constraints in the query, choosing more refined goals, thus retrieving more precise and shorter list of results or ii) eliminate some constraints from the query and look for more abstract goals or more undefined flow structure of the process, thus retrieving more results.

(i) The refinement is done mainly by the navigation inside sub-concepts, skipping instances connected to super-concepts. Since a concept can have many sub-concepts, the interaction with the user may be necessary to choose which path to follow. In Figure 6, an example of refinement is represented by a search for the goal “Creation Done”, which has at the moment no instances directly connected, however, its sub-concepts “User Profile Created” and “Catalog Entry Created” do have, and processes associated with these instances are results of a refined query.

(ii) If there are too few results, the framework can automatically search for instances connected to the parent concepts of the requested one in order to relax the query. For example in Figure 6, a query for processes related to the goal "Client Known" has just one instance as a result. If the framework decides to look for instances connected to "User Known", two instances match the query.

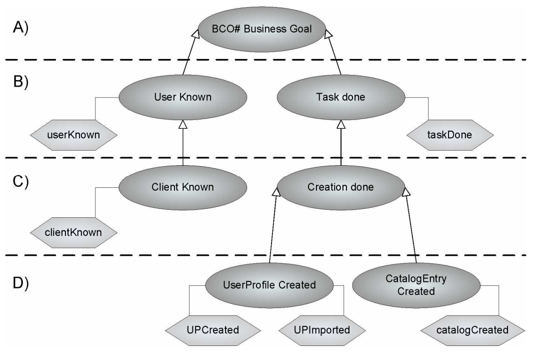


Figure 6: Business goals hierarchy

In general, when going deeper into the tree structure of goals one would retrieve fewer results (refinement), while when going upward in the tree structure the relaxation occurs.

A second way for relaxing a query is to "skip" target concepts in the query, e.g. using the "OR" statements. The user can specify which concepts are required and which are optional using the interface in Figure 4. Consider an example when the user wants models achieving goals A, B and C, where A and B are required to be parts of the model and C is optional. The framework can try to find models achieving all goals with the statement:

$$Q = \left\{ x \middle| x \in M \wedge ant\left(x, A\right) \wedge ant\left(x, B\right) \wedge ant\left(x, C\right) \right\}$$

Where *Q* represents the resulting set of process models, *M* denotes the set of all process models and relation *ant* denotes that a model *x* is annotated using the ontology concept *A*.

In case of no result, the framework decides to relax the query, executing the statement:

$$Q = \left\{ x \middle| x \in M \wedge ant\left(x, A\right) \wedge ant\left(x, B\right) \right\}$$

In the ideal case, the framework would submit the following statement, preferring to answer the user queries where all three goals are achieved:

$$Q = \left\{ x \middle| x \in M \wedge \left(ant(x,A) \wedge ant(x,B) \wedge ant(x,C)\right) \vee \left(ant(x,A) \wedge ant(x,B)\right) \right\}$$

Here we see a clear need for ranking: results that fulfil all goals are matching perfectly the user query, while results obtained for the relaxed query are fulfilling just a part of the user's need. At present, the ranking of query results is not supported by our framework. That will be a part of the future work.

5 Related Work

The significance of querying business processes has been acknowledged by BPMI[8] that launched a Business Process Query Language (BPQL) initiative [BPM]. However, no standard specification has been published yet.

In [Be06], the authors present a query language for querying business processes, BP-QL. The query language is designed based on the BPEL[9] standard and thus focuses on querying executable processes. Our work focuses on the reuse of higher level business knowledge, i.e. BP artefacts. In addition, our query specification language is more expressive in that apart from constraints on data and control flow, the user can specify additional properties of the models he wants to retrieve.

The approach presented in [MCZ04] discusses a process component model for process knowledge reuse. Here, the process component model is characterized only using static information (domain, function, performance, lifecycle). We defined a more refined notion of BP artefacts which can be reused in different stages of the presented modelling lifecycle. In addition, with this approach the user is not able to specify behavioural queries which is a more intuitive way of specifying his requests.

We have not seen other approaches addressing the problem of querying in business process modelling using a rich formal model for business processes.

6 Conclusion & Outlook

In this work we have presented a framework which enables expressive querying in business process modelling phase. The framework enables the business expert to have a quick and easy access to the library of process artefacts. We have illustrated three key usage scenarios showing the benefits of using our querying framework. Furthermore, we have developed a prototype of the querying framework, based on the Maestro BPM tool. Currently we are performing a use case study and the evaluation of the results is the very next task.

[8] Business Process Management Initiative, http://www.bpmi.org/
[9] Business Process Execution Language, http://www.ibm.com/developerworks/library/specification/ws-bpel/

As our next step, we plan to embed MWB in our framework and investigate more advanced notions of process equivalence based on the theory of bisimulation. In the long term, we aim to define similarity measures for process models with the purpose of quantifying the level of similarity between two models, which can be used in ranking of query results.

Bibliography

[AB06] Frank Puhlmann Anja Bog. A Tool for the Simulation of Pi-Calculus Systems. In *Open.BPM 2006: Geschäftsprozessmanagement mit Open Source-Technologien, Hamburg, Germany*, 2006.

[Be06] Catriel Beeri, Anat Eyal, Simon Kamenkovich, and Tova Milo. Querying Business Processes. In Umeshwar Dayal, Kyu-Young Whang, David B. Lomet, Gustavo Alonso, Guy M. Lohman, Martin L. Kersten, Sang Kyun Cha, and Young-Kuk Kim, editors, *VLDB*, pages 343–354. ACM, 2006.

[BPM] BPMI. Business Process Query Language. http://www.service-architecture.com/web-services/articles/business_proc% ess_query_language_bpql.html.

[CKO92] Bill Curtis, Marc I. Kellner, and Jim Over. Process Modeling. *Comm. of the ACM*, 35(9):75, September 1992.

[He05] Martin Hepp, Frank Leymann, John Domingue, Alexander Wahler, and Dieter Fensel. Semantic Business Process Management: A Vision Towards Using Semantic Web Services for Business Process Management. In Francis C. M. Lau, Hui Lei, Xiaofeng Meng, and Min Wang, editors, *ICEBE*, pages 535–540. IEEE Computer Society, 2005.

[JB96] Stefan Jablonski and Christoph Bussler. *Workflow Management: Modeling Concepts, Architecture, and Implementation*. International Thomson Computer Press, London, UK, 1996.

[MCZ04] Yujie Mou, Jian Cao, and Shen sheng Zhang. A Process Component Model for Enterprise Business Knowledge Reuse. In *IEEE SCC*, pages 409–412. IEEE Computer Society, 2004.

[MP07] Ivan Markovic and Alessandro Costa Pereira. Towards a Formal Framework for Reuse in Business Process Modeling. In *Workshop on Advances in Semantics for Web services (semantics4ws), in conjunction with BPM '07*, Brisbane, Australia, September 2007.

[Ma07] Ivan Markovic, Alessandro Costa Pereira, David de Francisco, and Henar Munoz. Querying in Business Process Modeling. In *2nd International Workshop on Business Oriented Aspects concerning Semantics and Methodologies in Service-oriented Computing, in conjunction with ICSOC '07, Sep 17-20, Vienna, Austria*, 2007.

[MPW89] Robin Milner, Joachim Parrow, and David Walker. A Calculus of Mobile Processes, Parts I and II. Technical Report -86, University of Edinburgh, June 1989.

[Pu06] Frank Puhlmann. A Tool Chain for Lazy Soundness. In *Business Process Management*, pages 9–16, 2006.

[Sa96] Davide Sangiorgi. A Theory of Bisimulation for the pi-Calculus. *ACTAINF: Acta Informatica*, 33, 1996.

[VM94] Björn Victor and Faron Moller. The Mobility Workbench — A Tool for the $$-Calculus. In David Dill, editor, *CAV'94: Computer Aided Verification*, volume 818 of *Lecture Notes in Computer Science*, pages 428–440. Springer-Verlag, 1994.

[YM94] Eric S. K. Yu and John Mylopoulos. Understanding "why" in software process modeling, analysis, and design. In *Proceedings of the 16th International Conference on Software Engineering*, pages 159–168. IEEE Computer Society Press, May 1994.