# Business process model repositories : framework and survey

# Business Process Model Repositories - Framework and Survey

Zhiqiang Yan, Remco Dijkman*, Paul Grefen

*Eindhoven University of Technology, PO Box 513, 5600 MB Eindhoven, The Netherlands*

**Abstract**

Large organizations often run hundreds or even thousands of business processes. Managing such large collections of business processes is a challenging task. Intelligent software can assist in that task by providing common repository functions such as storage, search and version management. They can also provide advanced functions that are specific for managing collections of process models, such as managing the consistency of public and private processes and extracting knowledge from existing processes to better design new processes. This paper presents a framework for repositories that assist in managing large collections of business process models, It also presents a survey of the functionality that existing repositories provide. The framework consists of a management model and a reference architecture. The management model lists the functionality that can be provided. The reference architecture presents the components that provide this functionality and their interconnections. The survey presents an overview of existing repositories and the functionality from the framework that they provide. By presenting a framework and a survey, this paper is a basis for a future research agenda. From the survey, we conclude that the field of Business Process Model Repositories is an important and active field in research and practice, but that complete repositories are not yet available and that existing repositories focus on traditional functionality rather than exploiting the full potential of information management tools.

*Key words:* Business Process Model, Repository, Reference Architecture

## 1. Introduction

As it becomes more common for organizations to describe their operations in terms of business processes, collections of business process models grow to contain hundreds or even thousands of business process models. For example, the SAP reference model contains over 600 business process models [42] and a collection of business process models for Dutch local government contains a similar number of business process models [21]. Managing such complex process landscapes is a difficult task. Typical issues arise, like: being able to find a particular process in a collection, managing different versions of processes and maintaining consistency when multiple people are editing the same process at the same

*Corresponding author (Email: r.m.dijkman@tue.nl, Phone: +31-40-2474370, Fax: +31-40-2432612 )

*Email addresses:* z.yan@tue.nl (Zhiqiang Yan), p.w.p.j.grefen@tue.nl (Paul Grefen)

time. In addition to that, the availability of a large collection of processes opens up new possibilities, like: extracting knowledge about the operations of the organization from the collection or re-using (best-practice) process fragments from the collection to design new processes.

As a reaction, software tools have been developed to help preform such tasks. These tools have been built as extensions of general database and repository systems. However, they have been specialized for storing business process models by using conceptual models, for example database schemas, that are process specific and by defining process specific interfaces. The interface could, for example, take the form of a Web service interface or an API that has operations like 'addProcess' and 'searchTask' and at which process models can be imported or exported in process specific interchange formats like EPML or PNML [35]. We refer to such repositories as BP Model Repositories, which we define as repositories that are structured according to a process-specific conceptual model and/or that have a process-specific interface. In addition to exploiting the functionality that is commonly provided by repository and database management systems [12, 39], BP Model Repositories provide functionality that is specific for repositories that contain business process models. Examples of process-specific functionality include: functionality to assist with lifecycle management of business processes, functionality to help maintain consistency between the private view on business processes (which is the view that organizations have internally on their business processes) and the public view on business processes (which is the view on those parts of business processes that companies want to make visible publicly), and functionality to assist with configuration management of business processes as they are composed of (certain versions of) sub-processes and tasks.

To provide an overview of the functionality currently provided by BP Model Repositories, this paper provides a framework and a survey of existing BP Model Repositories. The contribution of this paper is twofold. Firstly, it presents a framework for BP Model repositories, which consists of a management model and a reference architecture. The management model lists the functionality that can be provided by BP Model Repositories, while distinguishing between functionality that is provided by general repositories and database management systems [12, 39] and functionality that is specific for repositories that contain business process models. The reference architecture presents the components that provide this functionality and their interconnections. Both the management model and the reference architecture are developed by studying existing BP Model Repositories. Secondly, this paper compares existing BP Model Repositories using the framework. To the best of our knowledge, this is the first survey of BP Model Repositories.

The remainder of the paper is organized as follow. Section 2 presents the general BP Model Repository management model, which lists the functionality that can be provided by BP Model Repositories. Section 3 presents a reference architecture for BP Model Repositories. Section 4 presents a list of existing BP Model Repositories and briefly introduces each BP Model Repository and the functionality that it provides. Section 5 shows to what extent each project implements the functionality of the general BP Model Repository management model and the differences between the technologies that are used for implementation. Section 6 concludes the paper.

## 2. BP Model Repository Management Model

Although, by definition, BP Model Repositories have in common that they have a process-specific conceptual model or interface, they vary with respect to the form of that structure or interaction facility. Also, BP Model Repositories vary with respect to the functions that they provide. This section defines the possible forms of a BP Model Repository structure or interaction facility and the functions that a BP Model Repository can provide.

We consider a BP Model Repository as a specialized repository. According to Bernstein and Dayal [12], a repository is "a shared database of information about engineered artifacts produced or used by an enterprise". Consequently, it should provide common database management services for data model creation and adaption, data retrieval, enabling data views, integrity management, access management and state management. It should also provide services that are specific for managing objects as opposed to data in general: object checkout/checkin, version management, configuration management, notification management, context management and workflow management. The functionality for general repositories, as it is summarized by Bernstein and Dayal [12] and by Sagawa [39], can be specialized and extended to develop repositories that are specific for storing and managing business process models. We developed such an extension by taking the work of Bernstein and Dayal [12] and Sagawa [39] as a starting point and specializing and extending it, based on functionality that can be observed in existing BP Model Repositories, as they will be described in section 4.

The resulting BP Model Repository management model is shown in table 1. It consists of three parts: the process data model, the process function model and the process management model.

### *2.1. Process Data Model*

The process data model prescribes how business process models and related data can be fed to the BP Model Repository and how they are stored internally. It consists of the meta-model, the storage model and the index model.

The *meta-model* prescribes what information can be and must be stored in the BP Model Repository by defining the concepts that are used in the repository and the relations between those concepts. Each BP Model Repository supports potentially supports a large number of concepts. We classify those concepts by identifying the *process aspects* and the *process types* that are supported by a BP Model Repository. We distinguish the following process aspects.

- The activity aspect (A) contains concepts to describe the activities that are performed in the context of a process.
- The control-flow aspect (CF) contains concepts to describe the control-flow relations between activities.
- The data aspect (D) contains concepts to describe the information that is used and changed during the execution of a process.

Table 1: BP Model Repository Management Model

| | | |
|---|---|---|
| Process data model | Process meta model | Process aspect<br>Process type<br>Process notation |
| | Process storage model | External process data model<br>Internal process data model<br>Process related data model |
| | Process index model | Process classifications<br>Other process indices |
| Process function model | Storage functions | Create<br>Delete<br>Update<br>Import<br>Export |
| | Retrieval functions | Navigate<br>Search<br>Query |
| | Integration functions | [Depend on external tools] |
| Process management model | Process-specific management | Version management<br>Configuration management<br>Lifecycle management<br>Process view management |
| | General repository management | Access management<br>Integrity management<br>Transaction management<br>Checkin/out management<br>Dispatch management<br>Notification management<br>Context management |

- The resource aspect (R) contains concepts to describe physical resources that are required to execute (activities in) a process, including human resources.
- The authorization aspect (Au) contains concepts to describe who is authorized to perform which part of a process.
- The organization aspect (O) contains concepts to describe the organizational structure, as it consists of people and organizational units, related to a collection of processes.
- The strategic goals aspect (G) contains concepts to describe the hierarchy of strategic goals and to describe the relations of those goals to the processes that are meant to achieve them.

- The monitoring aspect (M) contains concepts to define how the performance of a process should be monitored.
- The management control (MC) aspect contains concepts to define the management controls that are implemented by (parts of) processes.

We distinguish the following process types.

- A company specific process (C) is a process that is designed by a specific company to describe its own operations.
- A reference process (Re) is an abstract and standard processes that can be reused and adapted to develop company specific processes. If a reference process contains pre-defined configuration options, it is also called a configurable reference process.
- A process pattern (P) is a partial process that describes a best practice summarized from former experience.
- A process instance (I), or case, is an execution of a process for a customer.
- Historical information (H) consists of logs that contain information about executions of the process instances.

The meta-model also prescribes how information that is stored in a BP Model Repository is presented to the end-user, by associating a notation with its concepts. For example, a BP Model Repository can store the information for activities and control-flow relations between those activities, but that information can be presented to the user in (structured) natural language, in a standardized graphical notation like EPC or BPMN, or in a proprietary graphical notation. It is also common for a BP Model Repository not to prescribe a notation, but focus solely on defining its conceptual model and/or interchange format.

The *storage model* prescribes how the original information about the process must be technically provided to the BP Model Repository (*external data model*) and how it must be internally stored by the BP Model Repository (*internal data model*). The external and the internal model can be the same, for example each process can be stored as an XML file that is also used to exchange the process between the BP Model Repository and related tools, or they can differ, for example processes can be exchanged using XML but stored in a relational database. Other than that process related data, which is data that is used by, but not part of, the processes can be stored in the repository. Process related data includes: descriptors of web services that are used by the processes and ontologies that are used to relate terms from different processes. For example, the IBM BPEL repository [47] also stores WSDL web-service descriptors.

The *index model* prescribes the indices that are kept for process models, to allow both the user and the repository manager itself to quickly browse or search the collection of processes. An index that is commonly used is a *classification* of process models in terms of the business functions for which they are available. For example, we can classify processes into processes

for: sales, procurement, production, finance and support. Subsequently, we can distinguish different classes of procurement processes, like procurement of product related materials and procurement of non-product related materials, etceteras.

*2.2. Process Function Model*

A BP Model Repository should support a series of basic functions to effectively manipulate the processes that it stores. We identify storage functions, retrieval functions and integration functions.

The *storage functions* are the functions to create, update and delete processes or parts of processes, by creating, updating or deleting instances of the concepts that are defined in the process meta model. In addition to that functions exist to import complete processes into the repository, using the interchange format from the external data model, and to export complete processes from the repository using that interchange format.

The *retrieval functions* can be used to obtain the required process according to some criteria. There are three methods for retrieving processes: navigate, query and search. Navigation is the method of manually scanning processes in a list, or by using a classification or some other index. Search provides the function to get processes that match criteria that are given as keywords. Query provides more advanced functions to specify search criteria using a query language, such as IPM-PQL [17] or BPMN-Q [4]. Queries and query languages can have a focus on one or more process aspects or process types. Awad distinguishes the following foci for process query languages [4]; languages that focus on retrieving (elements of) processes (company specific or reference), languages that focus on retrieving (elements of) process instances and languages that focus on retrieving (elements of) process execution history.

The *integration functions* can be used to integrate a process repository with external tools. The type of integration varies, depending on the types of tools with which the technical realization that is used to achieve the integration. In the BP Model Repositories that we have studied, we have observed integrations with the following types of tools.

- Process modeling tools, which can be used to visually create, retrieve, update and delete processes.
- Report generators, which can be used to generate reports about (monitoring information of) processes and their properties.
- Process analysis tools, which can be used to analyze correctness, selected properties or performance of processes.
- Workflow engines, which can be used to execute business processes by performing activities, or notifying human resources that activities must be performed, according to the order specified by the control-flow relations. When executing a business process a process instance is created and monitoring information is being generated.
- Collaboration tools, which can be used to establish business collaborations based on processes in the repository.

Within the BP Model Repositories that we studied there was no strict separation with respect to what is considered internal functionality of the repository and what is considered external functionality that can be integrated with the repository. For example, query tools have been proposed as external tools [4], but at the same time tools for establishing collaborations between organizations, based on their processes, have been proposed as internal parts of the repository [45]. We made the separation between internal and external functionality above, based on what we most frequently saw in the BP Model Repositories that we studied.

### *2.3. Process Management Model*

Advanced management functions can be subdivided into functions that are provided by general repositories and functions that are provided only by BP Model Repositories.

The *process specific management functions* are: version management, configuration management, lifecycle management and view management. Although version management, configuration management and view management are also general repository functions (or even general database functions in the case of view management), these functions have been specialized to meet process specific requirements. The version management function enables multiple versions of the same process or activity to be maintained. The configuration management function makes it possible to maintain the relation between (a version of) a process and the (versions of) subprocesses and activities that it consists of. Although version and configuration management are also a general repository functions, specialized functionality is added to support requirements in the context of BP Model Repositories. For example, when a process is being executed and a new version of that process or a part of that process is created, a decision must be made as to whether the new version will be put into effect for process instances that are already running or not and, if so, for which process instances. The lifecycle management function maintains the stage in its lifecycle that a process is currently in. For example, a process can be under design, validation and current. Depending on the stage that it is, some operations can be performed on it and others cannot. For example, a new version cannot be created of a version that is under design, nor can a process that is still under validation be executed. The view management function makes it possible to create multiple views on a process. Although view management is a general database function, specialized functionality is added to support requirements in the context of BP Model Repositories. For example, it is common to keep a private view on a process, which represents the process as it is performed inside an organization. At the same time a public view (also called service) can be provided of what the behavior of the process to the outside world will be like, therewith preserving company secrets of how services are internally implemented and not bothering clients with details that do not concern them. To support the generation of the public view from the private view and to keep the two consistent, BP Model Repository specific functionality is needed.

The *general repository management functions* are: access management, integrity management, transaction management, checkin/out management, dispatch management, notification management and context management. The access management function ensures that people only have access to the objects in the repository that they are authorized to

view. The integrity management function ensures that the repository cannot get into an inconsistent state. Transaction management ensures that multiple changes to a repository can be performed in a transactional manner (i.e.: either all at once or not at all). Checkin/out management allows a user to check-out objects from the repository, therewith locking them so others cannot change them, make the desired changes and then check them in again by releasing the lock. Optionally, multiple people can be allowed to check-out an object at the same time, in which case check-in management should ensure that changes that are made to the same object by multiple people are properly merged. Dispatch management makes it possible to associate a work-order with an object, such that it is forwarded to people in the order specified in the work-order along with notes about what these people have to do with the object. (This is usually called 'workflow management' in repositories. We call it dispatch management to avoid the confusion with workflow tools that execute the processes in a BP Model Repository.) Notification management enables notifications to be generated in case an object in the repository is changed. The context management function allows collections of repository objects, called 'contexts', (also called 'projects' or 'workspaces') to be created and manipulated. Contexts can be stored persistently.

## 3. BP Model Repository Reference Architecture

This section presents a reference architecture for BP Model Repositories. The reference architecture was developed by investigating the architectures that are proposed by the concrete BP Model Repositories that will be presented in the next section. In that respect it should be characterized as a descriptive reference architecture (describing the commonalities in the architectures of existing BP Model Repositories), rather than a prescriptive reference architecture (prescribing how a BP Model Repository architecture should be structured) [3].

Figure 1 shows the reference architecture. It has five layers: the presentation layer, the process repository management layer, the repository management layer, the database management layer and the storage layer.

The presentation layer provides GUIs for users to interact with a BP Model Repository, so the users can easily interact with the functions provided by the repository. Not all concrete BP Model Repositories have a presentation layer.

The process repository management layer provides repository functions that are specific for BP Model Repositories. The functions are described in detail in the previous section. Although general database management systems and repositories implement general functions, such as querying and checkin/checkout, most BP Model Repositories choose to implement these functions themselves, because this allows them to at least provide a facade that applies the functions specifically to processes instead of general repository objects or database tables. The functions may, or may not, use general functionality provided by a general database management system or repository to implement the BP Model Repository-specific functions.

The repository and database management layers provide the functions that are generally provided by repository and database management systems, respectively. Most BP Model Repositories that we studied do not distinguish the repository management layer from the

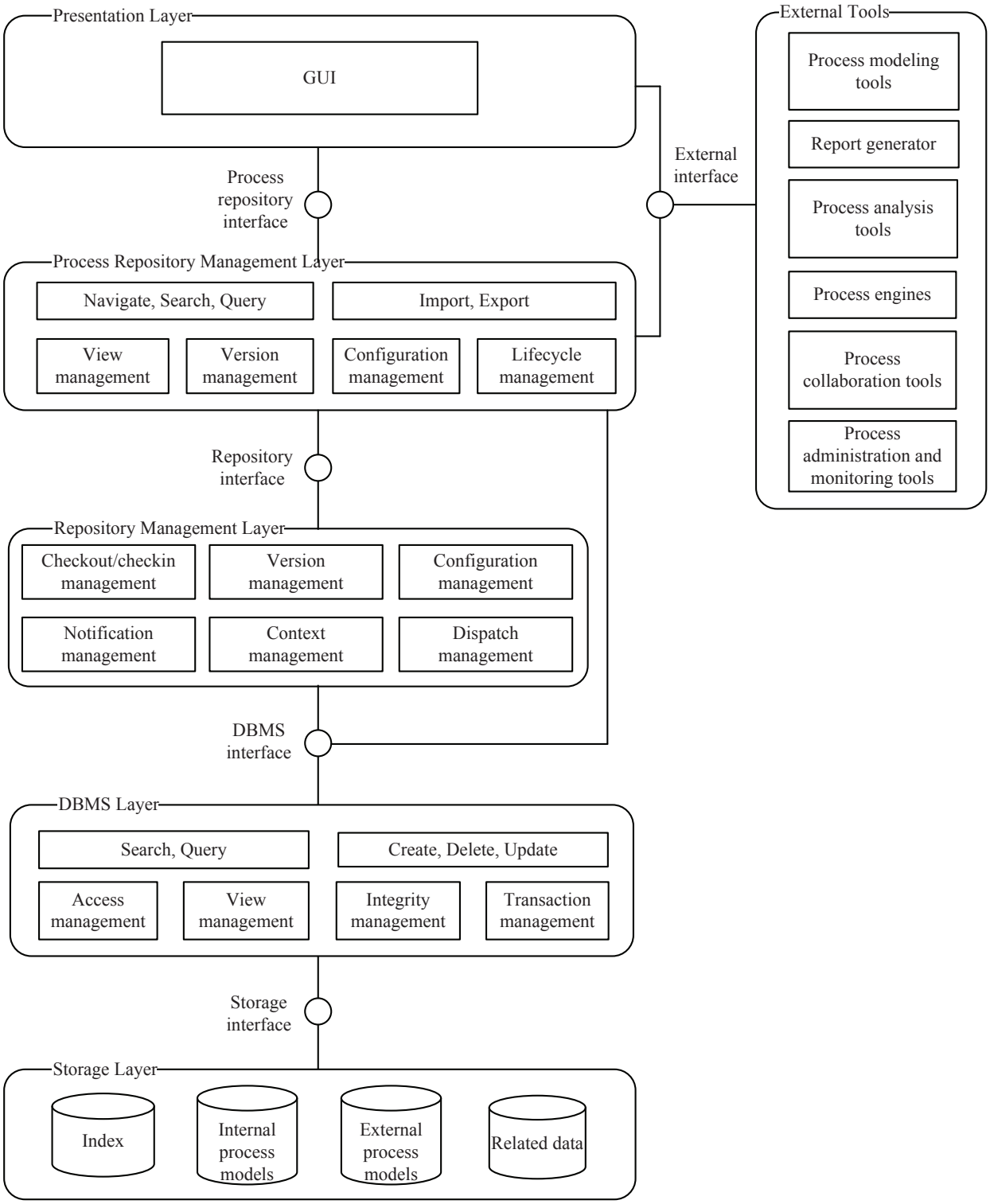


Figure 1: The BP Model Repository Reference Architecture

layer with process specific functionality. Instead, they will have a single layer that contains all management functionality. We introduce the distinction between the layers here, to clearly show that there is an architectural choice between implementing these layers in the process repository or obtaining them from general purpose repositories or database management systems.

The storage layer stores the process models, the related data and indices or classifications to enable fast querying, searching and navigation of the BP Model Repository. Process models can be stored both in an internal format, for example as rows in database tables, and in their original external format. In that case the internal format is used for fast and unified processing of process models in spite of their external format and the external format is used to maintain the relation with the original models. In most cases the storage layer is implemented by a general database management system. Relational, object-oriented and XML databases have all been observed in the concrete BP Model Repositories that were studied. Alternative implementations that have been observed are implementations using general repositories, of which one using a distributed repository, and an implementation in which the data is stored as files in a filesystem.

Well-defined interfaces should exist between the different layers. In most BP Model Repositories well-defined interfaces exist between the presentation layer and the process repository management layer and between the repository management layer and the database management layer. The technology that is used to implement the interfaces varies. The process repository interface can be implemented using a programming language API, but also using remote method invocation or even using web-services. The DBMS interface can be implemented using a (standard) API, but we have also observed concrete BP Model Repositories that added an additional layer that abstracts from the storage technology that was used, to allow different storage technology to be used without having to implement the repository functions.

In addition to the interfaces between the layers, interfaces can exist between the BP Model Repository and external tools. The presence and implementation of these interfaces varies largely. However they are all defined either to interact with the presentation layer, with the process repository management layer or with both. Interaction with the presentation layer enables the BP Model Repository to open a tool from the GUI of the BP Model Repository. Interaction with the process repository management layer enables an external tool to directly invoke the functions that are provided by the BP Model Repository.

## 4. Introduction of Existing BP Model Repositories

### *4.1. The process handbook*

The process handbook [44, 13, 31, 32] is a knowledge base of process descriptions. It consists of:

1 a classification scheme for organizing process models and their activities;

2 a collection of process model that is organized according to this classification; and

3 a collection of tools for managing the knowledge base.

The focus of the process handbook is on organizing knowledge about processes, not on providing detailed process models. Consequently, the repository is text-based rather than model-based and the meta-model mainly contains classes storing process models and their descriptions and for relating process models to each other and to their elements. Process models are provided to the process handbook through proprietary tools, in which the information entered in a text-based manner. Internally, the process models are stored in a general database.

The process handbook contains an elaborate classification scheme, which is organized on two dimensions: specialization and decomposition. Specialization allows specialized and, conversely, generalized versions of process models to be considered, like procurement can be specialized into procurement of non product-related goods. Decomposition allows an activity to be decomposed into parts, like procurement can be decomposed into drafting requirements, selecting vendors, . Along these two dimensions, a number of classifications are developed, such as a classification that decomposed operations into business functions and activities, which can subsequently be specialized for a particular industry or company.

Asides from standard functionality for storing and retrieving information about process models, the process handbook supports browsing the process models along the two dimensions. For each process model the user can browse generalizations, specializations, parts and wholes of which the process model is itself a part. In addition to that the process handbook supports text-based search.

*4.2. The process reuse architecture*

The process reuse architecture [23] is an architecture and prototype tool meant for storing reusable process models, either as a whole or in part.

The main process type that the process reuse architecture considers is the configurable reference process (which it calls process framework). To configure a reference process model, the process reuse architecture also stores process patterns and company specific process models. These can be reused either as a whole or in part to adapt a configurable reference process to the requirements of a particular organization. Like the process handbook, the process reuse architecture focuses on describing rather than modeling processes. However, the process reuse architecture has a more elaborate set of concepts for creating those descriptions, covering description of: process characteristics, organization structure and relation of process models to strategic goals and controls [24]. In addition to concepts for describing process models, the process reuse architecture uses reuse guidelines to describe configuration points in the reference architecture. The different types of process models can be stored in the database using an XML-based file format.

The process reuse architecture supports standard functionality for storing and retrieving process information. In addition to that it allows text-based search and navigation of the repository via three classifications. Process models and their elements are classified by means of the actions that they can perform and the objects on which they perform them. Actions are stored along with synonyms, allowing for more user-friendly search. Process patterns are

classified as individual (regular) patterns, pattern families, which represent a sets of patterns that can be used in combination to solve a larger problem and community patterns, which are patterns targeted towards a particular problem domain.

### *4.3. The library for process programming*

The library for process programming [51, 52] is an architecture for organizing business process models into libraries from which individual business process models can be reused.

The library for process programming focuses on the definition of processes, the enactment of process instances and adaptation of processes for reuse. For the purpose of defining processes, the system introduces a language called 'P', which resembles a textual programming language. It focuses on the definition of processes and activities, control-flow and data used by processes.

The system provides four mechanisms for reusing existing process definitions:

1 inheritance, which allows for a process to 'inherit' all activities and the control-flow from another process and extend or modify those activities and the control-flow;

2 nesting, which allows for a process to be reused integrally in another process;

3 integration, which allows for a process to reuse external resources, which are not necessarily processes; and

4 reflection, which allows for obtaining information about running instances and changing the process definition of running instances.

Inheritance and nesting of processes lead to tree structures that can be traversed. Such tree structures can be used to reveal the configuration of a process as it is composed of other processes and activities.

### *4.4. The repository for integrated process management*

The repository for integrated process management (IPM) [18, 16, 17] is a repository for managing business process models throughout their lifecycle. As such it does not only have basic functions for storing and retrieving business process models, but also advanced functions for version and configuration management.

Process information can be exchanged with the repository through the IPM Executable Process Definition Language (IPM-EPDL) [16], which is an XML-based format which can be used to store information about process models and activities, control-flow, organizational structure, authorization and resource assignment, data and monitoring. Internally, the repository has connectors for storing process information in an XML database or a relational database. IPM focuses on storing company specific process models and running instances of those processes.

The IPM repository has extensive support for storing and retrieving process models and related information. For that purpose a query language, the IPM Process Query Language (IPM-PQL) , has been defined, which has support for process-specific queries like searching

processes that have a certain activity or a certain transition from one activity to another. IPM also supports browsing processes in the repository using a number of classifications.

The IPM repository has two interfaces, an external interface that uses the IPM-EPDL to exchange process information with external tools such as modelling, analysis and enactment tools and an internal interface to connect the repository with a database.

The IPM repository has support for three management functions: lifecycle, versioning and configuration management. The lifecycle management function ensures that processes go through a number of states during their life, such as 'in design', 'validated' and 'in operation'. It also ensures that certain operations can only be performed on a process in certain states. For example, only validated processes can be enacted. Version management allows multiple versions of a process to be created and configuration management enables users to maintain relations between (versions of) processes and (versions of) the sub processes of which they are composed.

### *4.5. RepoX*

RepoX [43] is an XML-based process model repository, which is a part of the METEOR workflow system. Although it is part of the METEOR project, it has been specifically developed with the intention to standardize the exchange of process models between a process definition tool and a workflow engine (known as interface 1 of the workflow reference model [50]).

Process models can be exchanged with the repository, using a XML documents in a predefined format. Internally, the models are stored in an object-oriented database. RepoX stores the control-flow aspect of process models along with the data that is used in the processes and the roles that are authorized to perform tasks in the processes. Since RepoX is primarily meant as the repository for a workflow engine, it stores company-specific process models.

RepoX has functionality for version and configuration management. Versions of individual tasks, sub-processes a full processes can be stored in the repository, along with configuration information about which version of a task or sub-process is a part of which version of a process. RepoX also supports check-in and check-out of processes and their elements.

### *4.6. The repository for workflow systems*

In [27] a repository for workflow systems is described. The project provides a conceptual model for storing workflow models, requirements for managing workflow model repositories, and an architecture design for implementing a repository manager.

The repository supports the activity, control-flow, data, resource and monitoring aspects, and stores workflows in an object-oriented database. As defined in its workflow metamodel, it considers a workflow specification or resource as an entity, a workflow task as a weak entity, and constraints between tasks as relationships between entities. A workflow specification contains several workflow tasks, and constraints consist can be sequential composition of tasks, conditional choice between tasks, iterative performance of a task, split (and/or) between tasks and join (and/or) of tasks.

The Object-oriented Database Management System (OODBMS) provides the support the basic functionality for storing and retrieving workflows. Also, with help of scheduling, tracking and administration tools, the repository can record and manage monitoring information during the workflow runtime.

To support version and configuration managements, the repository defines a class named 'versionedObject' to record the version information, and the class also defines several functions to provide checkin/checkout and notification functions.

### *4.7. The BPMN repository architecture*

The BPMN repository architecture [45] is a repository architecture, rather than a complete implementation. It is specifically targeted towards BP Model Repositories that support inter-organizational processes, i.e.: processes or compositions of multiple processes that span organizational boundaries.

The BPMN Repository Architecture proposes that a broad range of process types and aspects should be supported for that purpose. Process aspects that should be supported are the control flow aspect, the data aspect, the organizational structure and authorization information, monitoring information and controls. Process types that should be supported are reference models, company specific models, process instances and historical information about process instances. The BPMN Repository Architecture prescribes that processes and related information should be stored in a standardized XML format. In particular it proposes the BPMN, along with a standardized BPMN XML interchange format, as the standardized set of concepts.

The BPMN Repository Architecture proposes four architectural elements that are specific for the support of inter-organizational processes. Firstly, it proposes that ontologies should be stored in the repository. Ontologies should facilitate the integration of processes from multiple parties, which use different terminology that can be translated using . Secondly, it proposes that a distinction is made between private repository information that is available only to a single party and public repository information that a party makes available to its partners. Thirdly, it proposes that a distributed repository is used, which corresponds to the distributed nature of inter-organizational processes. Fourthly, it proposes that parties that are involved in the processes can specify the processes in their own modeling notation of choice and that transformations are used from those notations into the common BPMN XML format.

### *4.8. Oryx*

Oryx [37, 19] is a web-based process modeling tool that supports users browsing, creating, storing and updating process models online. The tool uses a repository for storing the business process models that are created with it.

Oryx mainly focuses on the activity and control-flow aspect; it stores company-specific process models. It supports many (process) modeling notations, including: BPMN, EPC, Petri nets, Workflow nets, FMC Block Diagrams and XForms. Internally, processes are stored in a database; externally it represent process models in RDF format.

Oryx does not only provide basic storage functions (create, update and delete), but also implements import and export functions. For example, for the BPMN notation, Oryx can import processes from ERDF and JSON formats and processes can be transformed from BPEL. It can export process models in ERDF, JSON,RDF,PNML, XPDL, and XHTML formats or convert to Petri nets. For querying business processes, Oryx integrates BPMN-Q [4, 6, 7, 5] as part of the project. BPMN-Q is a visual query language for business process models, which extends the language elements of BPMN to support queries.

### *4.9. BP-Suite*

BP-Suite [11] is a tool suite based for execution of processes specified in BPEL. It consists of BP-QL (a query language for business process definitions) [8], BP-Mon (a tool for monitoring running business process instances) [9, 10, 11] and BP-Ex (a tool for analyzing the logs of the executions) [11].

BP-Suite supports the activity, control-flow, data, resource, organizational structure, monitoring and authorization aspects. It supports storing process models, process instances and historical information about process instances. Process models are stored in the BPEL XML format.

Asides from the standard database functionality for storing and retrieving process models, BP-Suite supports queries on all three types of processes that it can sotre (definitions, running instances and logs).

### *4.10. ProcessGene*

The ProcessGene project [48] provides a tool for querying business process models. It consists of four parts: a Scoping-Assistant (SA), a Query Specification Interface (QSI), a Query Interpreter (QI) and a Query Results Packager (QRP). Users of the ProcessGene provide querying scope and specifications by the SA and QSI; then the QI compiles specification to querying requirements and the QRP returns querying results.

ProcessGene focuses on the activity, control flow and authorization aspects and supports reference and company specific process models.

ProcessGene supports standard database functionality for storing and retrieving process models. In addition to that it is meant for querying business process models.

### *4.11. The Process Variants Repository*

The Process Variants Repository (PVR) [28, 29] provides mechanisms for dealing with different variants of a business process model at run time. For this purpose, it provides functionality for storing a (company-specific) business process model, along with (restrictions on) allowed variations of that business process model. Historical information about variations that are performed at run-time is also stored, such that it can be used to improve the business process model.

As defined in the process variant schema a variant consists a process model, an execution sequence based on the process model, resources, data, tasks, a design description and

authorizations. Accordingly, PVR supports the activity, control-flow, data, resource, monitoring and authorization aspects and stores company-specific processes, process instances and historical information about process instances.

Although PVR is designed for querying process variants (logs), it also provides an support for querying process definitions. In addition to that it provides standard functionality for storing and retrieving process models and their variants.

### *4.12. The Querying Framework*

The Querying Framework [33, 34] is a framework for developing advanced querying mechanisms for a business process repository.

The Querying Framework supports the activity, control-flow, data, goal, resource and authorization aspect. The resource aspect includes information resources. Hence, the data aspect is supported at a high level of abstraction. The Querying Framework stores information about these aspects in an RDF-based [14] ontology repository. The language that is used to enter the information is the WSML [15]. Information can be stored about process patterns, process fragments, company specific processes and, although they are not explicitly mentioned, reference processes.

The repository can be queried using WSML Logical Expressions. The Querying Framework has specific support for searching process fragments that complete a selected (incomplete) process model part and for searching substitutions for a selected process model part.

### *4.13. The Semantic Business Process Repository*

The Semantic Business Process Repository (SBPR) [30] is an ontology-based repository for storing business process models.

The SBPR does not commit to a particular set of aspects of business process models that must be stored. Instead, it requires that the repository is configured with a process ontology, of which the concrete process models are be instances. In particular, it considers BPMO [25], sBPMN [36], sEPC citeKeller92 and sBPEL [2] as ontologies. These ontologies must be specified in WSML. Internally, business process models are stored in a relational database.

Asides from the standard CRUD functionality, the SBPR supports querying, using WSML as a query language, versioning and check-in/check-out management.

### *4.14. The BPEL Repository*

The BPEL repository is a process repository, developed as an Eclipse plug-in, which is designed for storing and retrieving business processes in the BPEL format along with associated metadata [46**?** , 47].

The repository uses the BPEL XML format as its external (interchange) format and stores the process models and their elements internally as objects in an EMF repository.

Besides the standard functionality for storing and retrieving process models, the BPEL repository can interact with query engines that are built on the EMF repository (such as the Kent OCL engine), EMF extensions and other external software.

### *4.15. Prosero*

The Prosero project [22] combines business process management and Service Oriented Architecture (SOA) technology to support business process outsourcing based on Web Services.

The (semantic) repository of Prosero consists of four components: the terminology, the Reference Model Repository (RMR), the Web Service Repository (WSR) and Customer Model Repository (CMR). These components can be used to store reference process models and company-specific process models and process instances. Prosero has an external interface for modeling tools (organizational structure modeling, data modeling and business process modeling) and the ActiveBPEL4People engine. The aspects that are supported by Prosero are the activity, control-flow, data, resource, authorization and organizational structure aspect. Prosero stores BPMN processes and executes BPEL processes, consequently it has a BPEL generator, which can transform BPMN into BPEL.

Prosero support standard functionality for storing and retrieving process information. And with the help of the ActiveBPEL4People engine, Prosero can deploy, automate and analyze processes.

### *4.16. OSIRIS*

OSIRIS (Open Service Infrastructure for Reliable and Integrated process Support) [49, 40, 41] has been proposed for peer-to-peer process execution. The process repository support that it provides focuses on storing business process models, service specifications as they are provided or used by business processes and instances of executing business process models. In addition to that, OSIRIS provides supports typical peer-to-peer functions, such as concurrency control and load balancing.

OSIRIS supports the activity, control-flow and data aspect and stores company specific process models (or services) and running process instances. Internally, process models and service specifications are stored in a database.

In addition to standard functionality for storing and retrieving process models, OSIRIS has an interface with a modeling tool, named O'Grape, which uses a proprietary notation.

## 5. Comparison of Existing BP Model Repositories

In this section we compare the concrete BP Model Repositories that were presented in the previous section, using the framework that was explained in section 2 and 3.

### *5.1. The process data model*

Table 2 shows how the concrete BP Model Repositories implement the process data model. The aspects and model types that are supported by a BP Model Repository are identified by one or two letters as given in section 2.

The table shows that, except for the MIT Process Handbook, all concrete BP Model Repositories consider storage of the control-flow aspect. The MIT Process Handbook focuses on storing textual descriptions of the processes and the activities that occur within those processes; it shows the order in which activities usually occur in a process, but no

comprehensive control-flow. In addition to control-flow, the data and authorization aspect are frequently supported. There is a strong relation between whether or not the resource aspect is supported and whether or not a concrete BP Model Repository integrates with a process engine; out of the 7 BP Model Repositories that integrate with a process engine, only the Process Library does not support modeling the resource aspect. The relation between resource aspect support and engine integration is expected, because a process engine has to distribute work to resources. The Process Library relies on external tools to perform the distribution to resources; it focuses solely on the control-flow and data aspects. The control aspect is only supported by the Process Reuse Architecture and the BPMN Repository Architecture. Both these LPBRs also support the monitoring aspect and enable their users to relate monitoring information to controls, to relate values for performance indicators to controls.

Most tools focus on storing company-specific business process models. Two classes can be recognized for the BP Model Repositories that also support reference processes. The first class is the class of LPBRs that focus on storing reference processes to make them available as a knowledge base. The MIT Process Handbook falls into this class. The second class is the class of BP Model Repositories that focus on adapting reference processes to develop company-specific processes. The Process Reuse Architecure, the BPMN Repository, ProcessGene, the Querying Framework and Prosero fall into this class.

As external (interchange) format, most repositories either use XML or an XML-based standard (IPM-EPDL, BPEL, WSML, BPMO, sBPMN, sEPC or sBPEL). The exceptions are the MIT Process Handbook and the Process Library. The MIT Process Handbook is available only through a web-based user interface, through which users interact using natural language. The Process Library stores and exchanges its models through the P-Language, which strongly resembles a programming language. Internally, most BP Model Repositories store processes in a database. They either store them in their external format as 'blobs' or 'clobs' in a database or they store them in a more fine-graned manner, keeping separate tables or objects for separate elements (for example using a separate table for tasks, for processes and for data elements). Three BP Model Repositories, the BPMN Repository, the Querying Framework and the IBM BPEL Repository, use a general repository instead of a database. In the repository the processes are stored as repository objects. The process library stores processes as files in the file system. A few BP Model Repositories also store process-related information. This is limited to storing ontologies that establish a unified terminology and web service descriptors that define the web-services that are invoked by the processes in the repository.

As classifications, some BP Model Repositories allow their users to flexibly define process categories and classify the processes in these categories. Pre-defined classifications that have been proposed are a classification according to part-whole relations, a classification according to generalization-specialization relations, a classification of process patterns and a classification of process facets. Only 5 out of the total of 16 BP Model Repositories use a classification scheme.

Table 2: The process data model comparison

| | Process meta model | | | Process storage model | | | Process index model | |
|---|---|---|---|---|---|---|---|---|
| | aspect | type | notation | external | internal | related | classifications | others |
| MIT Process Handbook | A | Re | Structed natural language | Not specified | Database | | Part-Whole, Generalization-Specialization | |
| Process Reuse Architecture | A, CF, D, R, Au, G, O, MC, M | Re, C, P | Structed natural language | XML | Database | | Facets, Patterns | |
| Process library | A, CF, D | C | P language | P language | P language | | Part-Whole, Generalization-Specialization | |
| IPM | A, CF, D, R, O, M, Au | C | Proprietary (graphical) | IPM-EPDL | Database | | Categories | |
| RepoX | A, CF, D | C | Not specified | XML | Database (tables/ objects) | | | |
| Workflow repository | A, CF, D, R, M | C, I | Not specified | Not specified | Database | | | |
| BPMN Repository | A, CF, D, R, MC, O, M, Au | Re, C, I, H | BPMN | XML | Repository objects | Ontology | | |
| Oryx | A, CF | C | BPMN, EPC, Petri nets | RDF | Database | | | |
| BP-Suite | A, CF, D, R, O, M, Au | C, I, H | BPEL | BPEL | Database | | | |
| ProcessGene | A, CF, Au | Re, C | Not specified | Not specified | Database | | Categories | |
| PVR | A, CF, D, R, M, Au | C, I, H | Proprietary (graphical) | Not specified | Database | | | |
| Querying Framework | A, CF, D, G, R, Au | Re, C, P | Not specified | WSML | Repository Objects (RDF-like) | Ontology | | |

Table 2 – Continued

| | Process meta model | | | Process storage model | | | Process index model | |
|---|---|---|---|---|---|---|---|---|
| | aspect | type | notation | external | internal | related | classifications | others |
| SBPR | A, CF, D, Au, R | C | BPMO, sBPMN, sEPC, sBPEL | WSML | Database (tables) | | | |
| IBM BPEL Repository | A, CF, Au | C | BPEL | BPEL | Repository Objects (EMF) | WSDL | | |
| Prosero | A, CF, D, R, O, Au | Re, C, I | BPMN, BPEL | XML | Database | Terminology, WSDL | | |
| OSIRIS | A, CF, D | C, I | Proprietary | Not specified | Database | | | |

*5.2. The process function model*

Table 3 shows how the concrete BP Model Repositories implement the process function model.

Most BP Model Repositories support the create, update and delete storage functions. Exceptions are the MIT Process Handbook and the Process Library. Although, strictly speaking both BP Model Repositories do allow processes to be created, updated or deleted, they do not provide a public interface to do that. The MIT Process Handbook only allows for maintenance by certain people and the Process Library uses the file system for storage, but does not provide repository functionality for that purpose. We say that a BP Model Repository supports import and export of models if and only if they have an external file format. Files can then be imported or imported in that external file format.

Most BP Model Repositories support one or more functions to search, query or navigate the repository. Reading the papers in which the repositories are described leads to the conclusion that providing such functionality is an important motivation for developing a process repository. The support for such functionality is diverse. The table shows which type of functionality each BP Model Repository supports: search, query or navigation. It also shows for which types of process each BP Model Repository supports retrieval functionality. Retrieval functionality can be supported for process definitions, which includes definitions of reference processes, company-specific processes and process patterns, depending on the types of processes that the BP Model Repository supports. Retrieval functionality can also be supported for process instances and for historical information about process instances that have been running in the past. If applicable, the table shows the query language that is used by the repository. Interestingly, only IPM, BPMN-Q, BP-Suite and PVR provide a process-specific query language. The other BP Model Repositories use a general purpose query language. The Querying Framework supports a graphical manner for specifying queries by

selecting a part of a process model. Although Prosero and OSIRIS do not explicitly claim to support process search, they should be able to provide such functionality through the (SQL-based) search mechanisms that general databases provide.

Table 3: The process function model comparison

| | Process function model | | |
|---|---|---|---|
| | Storage | Retrieval | Integration |
| Process Handbook | Restricted | Search, Navigation of Definition | |
| Reuse Architecture | C,D,U,I,E | Search, Navigation of Definition | |
| Process library | File system | Search, Navigation of Definition | Engine |
| IPM | C,D,U,I,E | Query, Navigation of Definition (IPM-PQL) | Engine, Analysis |
| RepoX | C,D,U,I,E | Query, Navigation of Definition (SQL, XQuery) | |
| Workflow repository | C,D,U | Query of Definition, Instances (SQL) | Engine, Analysis |
| BPMN Repository | C,D,U,I,E | | Engine, Modeling, Collaboration |
| Oryx | C,D,U,I,E | Query of Definition (BPMN-Q) | Modeling |
| BP-Suite | C,D,U,I,E | Query of Definition , Instances, History (Proprietary) | Engine, Analysis, Monitoring |
| ProcessGene | C,D,U | Query of Definition (SQL) | |
| PVR | C,D,U | Query of Definition, Instances (Proprietary) | Engine, Analysis |
| Querying Framework | C,D,U,I,E | Query, Navigation of Definition (WSML) | Modeling |
| SBPR | C,D,U,I,E | Query of Definition (WSML) | Engine |
| BPEL Repository | C,D,U,I,E | Query of Definition (OCL) | Engine(OCL) |
| Prosero | C,D,U,I,E | | Modeling, Engine, Analysis |
| OSIRIS | C,D,U | | Modeling |

The level of integration of the BP Model Repositories with other tools is low, considering that BP Model Repositories are meant as supporting technology; they are only meant to store process models, process models must be designed in external tools and operations on them must also be performed by external tools. We attribute the lack of integration with external tools to the fact that most of the repositories are research prototypes, developed specifically to show the feasibility of and demonstrate the repository technology.

### *5.3. The process management model*

Table 4 shows how the concrete BP Model Repositories implement the process management model.

Table 4: The process management model comparison

| | Process management model | |
|---|---|---|
| | General repository | Process-specific |
| Process Handbook | DBMS | |
| Reuse Architecture | DBMS | |
| Process library | File system | |
| IPM | DBMS | Lifecycle, Versioning, Configuration |
| RepoX | DBMS | Versioning, Configuration |
| Workflow repository | DBMS | Versioning, Configuration |
| BPMN Repository | Distributed Repository | |
| Oryx | DBMS | |
| BP-Suite | DBMS | |
| ProcessGene | DBMS | |
| PVR | DBMS | |
| Querying Framework | Repository | |
| SBPR | DBMS | Versioning |
| BPEL Repository | Repository (EMF) | |
| Prosero | DBMS | |
| OSIRIS | DBMS | |

Except for the Process Library, all BP Model Repositories use a database management system or a general repository to support storage of processes. The table shows which type of system is used. Theoretically, this means that the management functions that are provided by these systems are available to the users of the BP Model Repository. However, in practice, additional effort may be required to make these functions available in a practical manner. For example, to make access control available in a practical manner for a database in which different types of process elements are stored in different tables, a single user must be given access to each relevant table to be allowed to create or read a single process. To make access control more practical in such cases security roles should be defined that package the access to all tables relevant to a process.

Process-specific management functions are rarely provided by the repositories. In particular process view management is not supported by any of the repositories

## 6. Conclusion

This paper defines a Business Process Model Repository (BP Model Repository) as a repository that is structured according to a process-specific conceptual model and/or has a process-specific interface. It presents a framework for BP Model Repositories, which consists of a list of functions that they can provide and a reference architecture that is an abstraction of the architectures that they observe. It also compares existing BP Model Repositories, using the framework.

Based on this comparison, we draw the following conclusions.

We observe three motivations with which the BP Model Repositories are developed. Invariably, the focus of a BP Model Repository is on functionality that matches the motivation with which it has been developed. Firstly, a repository may be proposed as a means for storing business process models and easily finding a process, among the stored processes, that has certain criteria. In this case the repository focuses on functionality for intelligent navigation, search or query of a collection of business process models. Secondly, a repository may be proposed as a means for storing business process models as well as their running instances. In this case it focuses on the means of interaction with the workflow engine that executes the process instances. Thirdly, a repository may be proposed as a base of process (reference) models, or parts thereof, which can be re-used to develop company-specific process models. In this case it focuses on re-use mechanisms, such as allowing pre-set configuration options in a process model and using ontologies to align terminology between processes. Independent of the motivation, collaborative editing of process models receives attention, along with version and configuration management functions.

Most repositories focus on providing traditional repository functionality. However, as a repository of business process models represents a large base of information, more intelligent functionality can easily be envisioned. Especially in combination with monitoring information that can be obtained from workflow engines, repositories can be a good basis for developing business intelligence tools. Examples of such tools include tools for: automated conformance checks [38], process mining [1] and difference analysis [20].

As business process modeling is become an important paradigm for describing and organizing enterprises, BP Model Repository technology becomes more and more important in practice too. The motivations that researchers use to position their BP Model Repositories studied in this paper seem to reflect the main uses of BP Model Repositories in practice. In practice, BP Model Repositories are used as repositories for process modeling tools (such as Aris) or as repositories for process modeling and execution tools (such as IBM WebSphere). Collections of re-usable (reference) business process models are also available in practice. Although the general motivation and focus of the repositories in research that are studied in this paper and repositories that are used in practice are similar, the repositories in research have more advanced functionality. In particular, we have not observed process query languages in practice. Also, the repositories in research have more advanced mechanisms for re-use; in practice processes can only be re-used by taking an existing (reference) process and modifying it.

When looking at the list of functions that can be provided by BP Model Repositories, there is a clear gap with respect to view management. View management is a function that is available in general database technology. However, when applying it to process repositories, specific functionality must be made available to keep process views consistent or to derive one view from the other. None of the BP Model Repositories implement such functionality.

Finally, the architecture that is proposed by the BP Model Repositories studied in this paper exists at a high level of abstraction, outlining only the major components and their interactions. In addition to that most BP Model Repositories do not re-use any of the existing functionality in general repositories and databases, with the exception of querying, which is done by mapping specialized business process model query languages to general query languages. This is reflected in the BP Model Architectures in which all functions are implemented in a single layer, where we separated process specific, repository specific and database specific functions. However, when looking at reference architectures for general repositories [39], a more fine-grained reference architecture should be developed also for process repositories. This reference architecture should at least discuss how process specific repository functions can use general repository and database functions, provide more details with respect to the components that a process repository is composed of and incorporate details about how process-specific view management can be supported.

## References


[1] W.M.P. van der Aalst, and A. J. Weijters. Process mining: a research agenda. *Computers in Industry*, 53(3): 231-244, 2004.

[2] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. Business Process Execution Language for Web Services (Version 1.1), 2003.

[3] S. Angelov, P. Grefen, and D. Greefhorst. A Classification of Software Reference Architectures: Analyzing Their Success and Effectiveness. In *Proceedings Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture 2009*, Cambridge, UK, 2009.

[4] A. Awad. BPMN-Q: A language to query business processes. In: *Proceedings of EMISA 2007*, Nanjing, China, pp. 115–128, 2007.

[5] A. Awad, G. Decker, and M. Weske. Efficient compliance checking using bpmn-q and temporal logic. In *Proceedings of BPM 2008*, Milan, Italy, pp. 326–341, 2008.
[6] A. Awad, A. Polyvyanyy, and M. Weske. Semantic querying of business process models. In *Proceedings of EDOC 2008*, München, Germany, pp. 85–94, 2008.
[7] A. Awad, and F. Puhlmann. Structural detection of deadlocks in business process models. In *Proceedings of BIS 2008*, Innsbruck, Austria, pp. 239–250, 2008.
[8] C. Beeri, A. Eyal, S. Kamenkovich, and T. Milo. Querying business processes. In *Proceedings of VLDB 2006*, Seoul, Korea, pp. 343–354, 2006.
[9] C. Beeri, A. Eyal, T. Milo, and A. Pilberg. Monitoring business processes with queries. In *Proceedings of VLDB 2007*, Vienna, Austria, pp. 603–614, 2007.
[10] C. Beeri, A. Eyal, T. Milo, and A. Pilberg. Query-based monitoring of bpel business processes. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, Beijing, China, pp. 1122–1124, 2007.
[11] C. Beeri, A. Eyal, T. Milo, and A. Pilberg. Bp-mon: Query-based monitoring of bpel business processes. *SIGMOD Record*, 37(1):21–24, 2008.
[12] P.A. Bernstein, and U. Dayal. An overview of repository technology. In *Proceedings of VLDB 1994*, Santiago de Chile, Chile, pp. 707–713, 1994.
[13] P.A. Bernstein, C. Dellarocas, T.W. Malone, and J. Quimby. Software tools for a process handbook. *IEEE Data Engineering Bulletin*, 18(1):41–48, 1995.
[14] D. Brickley, and R.V. Guha. RDF vocabulary description language 1.0 (RDF schema). W3C Recommendation , 2004.
[15] J. de Bruijn (editor). The Web Service Modeling Language (WSML). WSMO Final Draft D16.v0.21, 2005.
[16] I. Choi, H. Jung, M. Song, and Y. Eyu. Ipm-epdl: an xml-based executable process definition language. *Computers in Industry*, 56(1):85–104, 2005.
[17] I. Choi, K. Kim, and M. Jang. An xml-based process repository and process query language for integrated process management. *Knowledge and Process Management*, 14(4):303–316, 2007.
[18] I. Choi, M. Song, C. Park, and N. Park. An xml-based process definition language for integrated process management. *Computers in Industry*, 50(1):85–102, 2003.
[19] G. Decker, H. Overdick, and M. Weske. Oryx C An Open Modeling Platform for the BPM Community. In *Proceedings of BPM 2008*, Milan, Italy, pp. 382–385, 2008.
[20] R.M. Dijkman. Diagnosing Differences between Business Process Models. In: *Proceedings of BPM 2008*, Milan, Italy, pp. 261–277, 2008.
[21] Documentair structuurplan. Retrieved February 20, 2009 from: http://www.model-dsp.nl.
[22] M. Elhadad, and M. Balaban. Effective business process outsourcing: The prosero approach. *International Journal of Interoperability in Business Information Systems*, 3(1), 2008.
[23] S. Fiorini, J. Leite, and C. Lucena. Process reuse architecture. In *Proceedings of CAiSE 2001*, Interlaken, Switzerland, pp. 284–298, 2001.
[24] S.T. Fiorini, J.C.S. do Prada Leite, and T.D.L.v.A. de Macedo-Soares. Integrating business processes with requirements elicitation. In *Proceedings of WETICE 1996*, Stanford, CA, USA, pp. 226–231, 1996.
[25] M. Hepp, and D. Roman. An ontology framework for semantic business process management. In *Proceedings of Wirtschaftsinformatik 2007*, 2007.
[26] G. Keller, G. Nüttgens, and A.-W. Scheer. Semantische Prozeẅmodellierung auf der Grundlage Ereignisgesteuerter Prozeẅketten (EPK). Veröffentlichungen des Instituts für Wirtschaftsinformatik, 1992.
[27] C. Liu, X. Lin, X. Zhou, and M. Orlowska. Building a repository for workflow systems. In *Proceedings of Technology of Object-Oriented Languages and Systems*, Nanjing, China, pp. 348–357, 1999.
[28] R. Lu, and S. Sadiq. Managing process variants as an information resource. In *Proceedings of BPM 2006*, Vienna, Austria, pp. 426–431, 2006.
[29] R. Lu, S. Sadiq, and G. Governatori. On managing business processes variants. *Data & Knowledge Engineering*, 68(7): 642–664, 2009.
[30] Z. Ma, B. Wetzstein, D. Anicic, and S. Heymans. Semantic business process repository. In *Proceedings*

*of SBPM 2007*, Innsbruck, Austria, pp. 92–100, 2007.

[31] T.W. Malone, K. Crowston, and G.A. Herman. *Organizing Business Knowledge: The MIT Process Handbook*. MIT Press, 2003.

[32] T.W. Malone, K. Crowston, J. Lee, B. Pentland, C. Dellarocas, G. Wyner, J. Quimby, C.S. Osborn, P.A. Bernstein, G. Herman, M. Klein, and E. O'Donnell. Tools for inventing organizations: toward a handbook of organizational processes. *Management Science*, 45(3):425–443, 1999.

[33] I. Markovic, and A. Pereira. Towards a formal framework for reuse in business process modeling. In *Proceedings of the BPM 2007 Workshops*, Brisbane, Australia, pp. 484–495, 2007.

[34] I. Markovic, A. Pereira, and N. Stojanovic. A framework for qureying in business process modeling. In *Multikonferenz Wirtschaftsinformatik*, 2008.

[35] J. Mendling, G. Neumann, and M. Nüttgens A Comparison of XML Interchange Formats for Business Process Modelling. In *Proceedings of EMISA 2004*, Luxembourg, Luxembourg, pp. 129-140, 2004.

[36] Object Management Group. Business Process Modeling Notation (BPMN) Version 1.1. OMG Final Adopted Specification formal/2008-01-17. Object Management Group, 2006.

[37] Oryx. 2008. Retrieved October 7, 2009 from: http://bpt.hpi.uni-potsdam.de/Oryx/WebHome.

[38] A. Rozinat, and W. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1): 64-95, 2008.

[39] J.M. Sagawa. Repository manager technology. *IBM Systems Journal*, 29(2):209–227, 1990.

[40] C. Schuler, R. Weber, H. Schuldt, and H. Schek. Peer-to-peer process execution with osiris. In *Proceedings of ICSOC 2003*, Trento, Italy, pp. 483–498, 2003.

[41] C. Schuler, R. Weber, H. Schuldt, and H. Schek. Scalable peer-to-peer process management - the osiris approach. In *Proceedings of ICWS 2004*, San Diego, California, 2004.

[42] T.A. Curran, and G. Keller. SAP R/3 Business Blueprint - Business Engineering mit den R/3-Referenzprozessen. Addison-Wesley, Bonn, Germany, 1999.

[43] M. Song, J.A. Miller, and I.B. Arpinar. RepoX: An XML Repository for Workflow Designs and Specifications. Technical Report UGA-CS-LSDIS-TR-01-012, University of Georgia, USA, 2001.

[44] The MIT process handbook project. 2003. Retrieved October 7, 2009 from: http://ccs.mit.edu/ph.

[45] T. Theling, J. Zwicker, P. Loos, and D. Vanderhaeghen. An architecture for collaborative scenarios applying a common bpmn-repository. In *Proceedings of DAIS 2005*, Athens, Greece, pp. 169–180, 2005.

[46] J. Vanhatalo. Building and querying a repository of bpel process specifications. Master's thesis, Helsinki University of Technology, Institute Eurecom and University of Nice-Sophia Antipolis, 2004.

[47] J. Vanhatalo, J. Koehler, and F. Leymann. Repository for business processes and arbitrary associated metadata. In *Proceedings of BPM 2006*, Vienna, Austria, pp. 25–31, 2006.

[48] A. Wasser, M. Lincoln, and R. Karni. Processgene query- a tool for querying the content layer of business process models. In *Proceedings of BPM 2006*, Vienna, Austria, pp. 1–8, 2006.

[49] R. Weber, C. Schuler, P. Neukomm, and H. Schuldt. Web service composition with o'grape and osiris. In *Proceedings of VLDB 2003*, Berlin, Germany, 2003.

[50] Workflow Management Coalition. The Workflow Reference Model (version 1.1). Technical Report TC00-1003, Workflow Management Coalition, 1995.

[51] G. Yang. Towards a library for process programming. In *Proceedings of BPM 2003*, Eindhoven, The Netherlands, pp. 120–135, 2003.

[52] G. Yang. Process library. *Data & Knowledge Engineering*, 50:35–62, 2004.