APROMORE: An Advanced Process Model Repository

Marcello La Rosa[*,a], Hajo A. Reijers[b], Wil M.P. van der Aalst[b], Remco M. Dijkman[b], Jan Mendling[c], Marlon Dumas[d], Luciano García-Bañuelos[d]

[a] *Queensland University of Technology, GPO Box 2434, Brisbane 4001, Australia*
[b] *Eindhoven University of Technology, PO Box 513, 5600 MB Eindhoven, The Netherlands*
[c] *Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany*
[d] *University of Tartu, J Liivi 2, Tartu 50409, Estonia*

Abstract

Business process models are becoming available in large numbers due to their widespread use in many industrial applications such as enterprise and quality engineering projects. On the one hand, this raises a challenge as to their proper management: How can it be ensured that the proper process model is always available to the interested stakeholder? On the other hand, the richness of a large set of process models also offers opportunities, for example with respect to the re-use of existing model parts for new models. This paper describes the functionalities and architecture of an advanced process model repository, named APROMORE. This tool brings together a rich set of features for the analysis, management and usage of large sets of process models, drawing from state-of-the art research in the field of process modeling. A prototype of the platform is presented in this paper, demonstrating its feasibility, as well as an outlook on the further development of APROMORE.

Key words: repository, business process model, process collection.

1. Introduction

Business process modeling has become a very popular form of conceptual modeling (Davies et al., 2006). A process model describes, often in some graphical notation, how a certain procedure is composed out of different tasks, which resources are involved in carrying out these tasks, and which objects are being manipulated (Curtis et al., 1992; Giaglis, 2001). One can roughly distinguish between process models that describe procedures as they exist (e.g., to show compliance with quality standards), or that capture alternative ways to produce a particular product or service (e.g., as blueprints for improvement projects). A process model can be used both within the specific context of IT deployment and for more business-oriented purposes (Bandara et al., 2005). Respective examples for these two types of process model usage are the configuration of a workflow management system, and the support of an activity-based calculation of a product's cost price.

The broad application of process modeling has stimulated contemporary organizations to create dozens, hundreds, and even thousands of process models (Becker et al., 2000; Gulla and Brasethvik, 2000; Reijers et al., 2009; Siegeris and Grasl, 2008). With such massive collections of of process models, an apparent issue is how to sensibly deal with these, in particular when considering that models may need to be consulted, updated, and re-used over longer periods of time by various stakeholders. This paper proposes an architecture for an Advanced Process Model Repository – APROMORE – which offers a rich set of features to maintain, analyze, and exploit the content of process models. The features that we envision go well beyond the data-management oriented functionalities that are offered by commercial process repositories. Instead, the emphasis is on sophisticated, state-of-the art functionalities such as advanced model-based analysis, comparison, unification, and consolidation. These functionalities may operate on *separate* and/or *sets of related* process models.

APROMORE has been implemented as an open-source SaaS (Software-as-a-Service). It is thought to be of interest to practitioners who wish to extract greater value from their process models' content, technology vendors that wish to extend their offerings by tapping into APROMORE's features, and researchers who wish to benefit from synergies by incorporating their techniques in the platform and re-using available techniques.

The structure of this paper is as follows. Section 2 provides a background on data repository technologies and specifically on process model repositories. This paves the way for Section 3 and Section 4, which respectively describe the envisioned main features of APROMORE and the service-oriented architecture to support and realize these features. Section 5 presents the internal process definition adopted by APROMORE, which is essential to deal with the multitude of process modeling notations. Section 6 provides a glimpse of the current prototype implementation and describes two typical application scenarios

[*]Corresponding author. Tel.: +61731389482; fax: +61731389390.
Email addresses: m.larosa@qut.edu.au (Marcello La Rosa), h.a.reijers@tue.nl (Hajo A. Reijers), w.m.p.v.d.aalst@tue.nl (Wil M.P. van der Aalst), r.m.dijkman@tue.nl (Remco M. Dijkman), jan.mendling@wiwi.hu-berlin.de (Jan Mendling), marlon.dumas@ut.ee (Marlon Dumas), lgbanuelos@gmail.com (Luciano García-Bañuelos)

that are supported by this implementation. Finally, Section 7 concludes the paper with a summary and an outlook on future work in this area.

2. Background

This section discusses the background of advanced process model repositories. In Section 2.1 we present concepts and solutions related to general data repositories. In Section 2.2 we then focus on commercial and academic process model repositories.

2.1. Data Repositories

Data repositories have been discussed for quite some time in the database research community. The term repository in this context refers to an extension of a database management system with an explicit control layer with a strong emphasis on metadata management. A repository can then be defined as a "shared database of information about engineered artifacts produced and used by an enterprise" (Bernstein and Dayal, 1994). Database repositories are closely interrelated with the management of static data models. Model management addresses challenges in this area on different levels, from representational questions on a structural level, to processing issues, and topics of organizational embeddedness in the socio-technical system of an enterprise (Dolk and Konsynski, 1984; Blanning, 1993). The main concepts of model management are models and mappings between models (Bernstein, 2003). The major share of mappings can be related to the areas of data warehousing (Jarke et al., 1999) and schema integration (Melnik et al., 2003). Research in all these areas is well-established for static data models, but an overarching approach for process model repositories with integrated model management functionalities is still missing these days.

2.2. Process Model Repositories

Process model repositories have been designed both with a focus on workflow execution and on conceptual modeling. In an execution environment, the major focus is on the provision of features for the definition of process control-flow, data structures, resources, and program interfaces (Leymann and Altenhuber, 1994). These main aspects are also present in contemporary BPM tools, but extended e.g. with discovery components for dynamic composition (Weber et al., 2007). In contrast, the focus of conceptual modeling frameworks is more on extension features. For example, IDS Scheer's ARIS process modeling tool supports the extension of the process metamodel along with customization of symbols (Scheer and Nüttgens, 2000)[1]. Standard features include model creation, modification and deletion, accompanied by simple lexical search (e.g. search all models containing 'Shipment Payment') and reporting functionalities (Lee and Joung, 2000). Similar functionalities are offered by ADONIS (Karagiannis and Kühn, 2002). Other solutions, such as Lombardi's Blueprint[2] and the ARIS Governance Engine, allow users to visualize changes between multiple versions of a process model. All these systems also offer access control. Specific requirements on a distributed environment including access rights are discussed in (Theling et al., 2005).

The application of semantic technologies has been considered from various angles for process repositories. Work on the commercial tool Semtalk has early recognized the potential of formal ontologies for adding and dynamically changing metamodels in a process repository (Fillies et al., 2003). Formal ontologies are also used in the work of Klein and Bernstein (2004). The authors utilize extensive metadata in the process repository for reasoning and process model retrieval. For querying process repositories, they define the Process Query Language (PQL) (Klein and Bernstein, 2004). Repositories also play an important role for an overarching concept of Semantic Business Process Management (SBPM) (Hepp et al., 2005). In (Karastoyanova et al., 2008), the authors identify modeling, system configuration, execution and analysis as key features of a respective SBPM architecture. This architecture builds on three layers: persistence layer, service layer (including locking, version control, and rule inference), and a repository API on top.

3. Capabilities of an advanced process model repository

Besides the standard repository functionalities currently offered by commercial and academic products, such as check-in/check-out, access control, simple search queries, there are several functional areas that can be envisioned when considering *advanced* support for dealing with process model collections. In this section, we distinguish four areas to discuss such advanced features: i) *evaluation*, ii) *comparison*, iii) *management*, and iv) *presentation*, and use this classification to propose the capabilities of APROMORE, as reported in Table 1.

As will be shown, this proposal builds on a large set of existing contributions in terms of approaches and techniques which can be adapted to be incorporated in APROMORE. The existence of the various approaches and techniques demonstrates the potential power of APROMORE. It should be noted, however, that each of these contributions focus only on a small piece of the overall landscape of functionalities we envision, and tend to look at process models *in isolation*, rather than looking at a process model *in relation* to other models. Moreover, the majority of these approaches and techniques has been devised to work for specific process modeling languages only. Our goal is to provide a wide range of model-independent functionalities that can be directly applied to contemporary process notations such as BPMN, EPCs, YAWL, WS-BPEL, WF-nets, Protos, etc.

3.1. Evaluation

Evaluation is concerned with establishing the adherence of process models to various quality notions. This area includes

[1] www.aris.com

[2] www.lombardisoftware.com/bpm-blueprint-product.php

Evaluation	Comparison	Management	Presentation
Quality analysis	Similarity search	Harmonization-driven creation	Abstraction
Correctness analysis	Conformance analysis	Pattern-based creation	Secondary Notation
Performance analysis	Pattern-based analysis	Reference-based completion	Reporting
		Pattern-based completion	
		Individualization	
		Extension control	

Table 1: Classification of advanced process model repository functionalities according to service areas.

three topics: correctness analysis, performance analysis, and usability analysis. There is a rich body of knowledge that discusses *correctness* properties like liveness, boundedness or soundness, mainly based on Petri net concepts (Murata, 1989; Aalst, 1996; Hee et al., 2006). Empirical research has shown that process model collections from practice typically include a substantial rate of error models (Mendling, 2009). As shown in (Fahland et al., 2009) there are mature verification approaches available. However, these are rarely supported by commercial tools.

Performance analysis is also a well-established discipline with its roots in operations research and operations management (Anupindi et al., 1999). However, it is often impractical to get meaningful durations, data on execution times and probabilities of alternative branches for performance analysis. Moreover, existing models of human behavior in organizations are too simplistic as demonstrated in Aalst et al. (2010). Recent research derives simulation parameters from operational systems using process mining techniques (Rozinat et al., 2008). Such features are still missing in commercial tools.

The evaluation of process models has become subject to *usability* considerations. Research on process model understanding aims to identify the factors that foster or impede *model quality* (Mendling et al., 2007). Structural metrics such as size or complexity have proven to be closely connected with understanding and error probability (Mendling, 2008). Process modeling guidelines such as the Seven Process Modeling Guidelines (7PMG) (Mendling et al., 2009b) or (Becker et al., 2000) have been proposed. Future tools might directly support them by keeping track of complexity metrics.

3.2. Comparison

Comparison offers capabilities to determine similarities between models or identify relevant patterns. This area covers the topics of similarity search, conformance analysis, patterns-based analysis and extension analysis.

The heterogeneous representation of comparable behavior has raised the issue of *similarity* calculation (Dijkman, 2007; Dijkman et al., 2009). In essence, process model similarity determines how close the behavior of two process models is. It can be associated with syntactical, semantical, and contextual aspects of activities in a process model (van Dongen et al., 2008). Taking process behavior into account yields better results than classical metadata based process query techniques such as (Momotko and Subieta, 2004; Klein and Bernstein, 2004). Query languages such as BPMN-Q (Awad et al., 2008) can use similarity calculations for ranking query results.

The calculation of similarity is on a conceptual level closely related to checking whether certain patterns or process fragments are contained in process models (*pattern-based analysis*). Dedicated query languages such as BPMN-Q or PQL (Klein and Bernstein, 2004) support the formulation of queries to find such patterns.

Conformance analysis evaluates to which extent an input model conforms to a reference process model in a given domain. Respective research is discussed in (de Medeiros et al., 2008).

3.3. Management

Management refers to different ways to create, modify and complete process models, potentially based on existing content.

Harmonization-driven creation uses merging algorithms to create a new process from a set of similar models (which can be retrieved via similarity analysis). Related work has discussed process model integration techniques (Aalst and Basten, 2001; Preuner et al., 2001; Grossmann et al., 2005; Mendling and Simon, 2006; Gottschalk et al., 2008), but these techniques are still embryonic and only work for specific process modeling languages. Moreover, commercial tool support is still missing. *Pattern-based creation* allows one to create a process model based on the composition of a set of process fragments (so called "business patterns") for a specific domain. The general idea of creating a process model based on predefined building blocks has been presented in (Thom et al., 2007).

Individualization relates to the area of configurable process models (Rosemann and Aalst, 2007; La Rosa, 2009). It builds on algorithms to derive a correct process model from a configured process model (Aalst et al., 2009), potentially covering control flow, data and resources involved in a process (La Rosa et al., 2008).

Over time new versions of reference models may be shipped, e.g. resulting from bug-fixes or changes to legislative rules. Similarly, a reference model individualization may also be extended, e.g. to cover customer requirements that are not captured by the reference process model. In this context, *extension control* is required. It has to establish extension points to control the evolution of configurable reference models and of their individualizations, such that the two types of models can be kept in synchronization. This idea is illustrated in (Balko et al., 2009) but it has not yet been implemented.

3.4. Presentation

Presentation provides support for improving the understanding of large process models and collections thereof. It relates to useful abstraction mechanisms, secondary notations (color, size, etc.), and reporting facilities.

Abstraction is an important concept to achieve a task-oriented presentation of content for a particular user of a process repository. Different abstraction concepts such removing infrequent paths and activities and automatically collapsing nodes based on high cohesion/low coupling strategies have been proposed for the simplification and understandability of process models (Günther and Aalst, 2007; Streit et al., 2005; Eshuis and Grefen, 2008; Polyvyanyy et al., 2008).

Secondary notation (Green and Petre, 1996) is a powerful tool to emphasize relevant information without touching the formal structure of a process model. It includes the use of color palettes, e.g. by highlighting the most followed process flow depending on a given user context (Aalst, 2009), icons (Mendling et al., 2009a), and the change of the graph layout. The importance of graph layout is well understood in the conceptual modeling area (Ware et al., 2002; Schrepfer et al., 2009). Specific layout requirements of BPMN (Object Management Group, 2008) models have been recently discussed in (Kitzmann et al., 2009; Effinger et al., 2009).

Finally, *reporting* provides a range of model statics such as number of users or frequency of decisions, to accompany the more traditional visual representation of process models.

The four functional areas that have been discussed up to this point – evaluation, comparison, management, and presentation – characterize the main types of functionalities that can be offered by an advanced process model repository. We foresee scenarios where end users will be combining elements from different functional areas. For example, the result of a process model *evaluation* could lead to an improvement plan describing a number of modifications on the process model (*management*) to align the latter to a reference model (*comparison*). Or more, after *evaluating* the quality of a collection of process models, the best performing models are selected and *compared* to each other in order to detect similarities. This result is used to merge the selected models into a configurable reference model (*management*), which is then *presented* to the user via a combination of abstraction techniques.

In the next section, we describe the architecture of APROMORE which is tailored towards supporting the functionalities mentioned.

4. Architecture

We propose to implement APROMORE via a service-oriented architecture as illustrated in Figure 1. This architecture follows a three-tier model composed of an enterprise layer, an intermediary layer and a basic layer. The enterprise layer is the front-end of the repository. It hosts the *repository manager* – a public service which exposes the typical amenities of a repository, such as check-in/check-out, simple querying, views, version control, change notification, context management and security (Bernstein and Dayal, 1994). This service is the unique entry point to the repository and can be accessed directly by BPM Suites (BPMS) or reference model vendors for cross-enterprise integration, or via a Web portal by the users of an organization.

The basic layer encapsulates the business logic and data of a traditional software architecture. The business logic consists of the algorithms to operate over process model collections, e.g. matching algorithms, merging algorithms, individualization algorithms. These algorithms are needed to provide the advanced functionalities described in Section 3. Each class of algorithms (evaluation, comparison, management and presentation) is encapsulated by a logic-centric service for reusability and maintainability purposes.

The repository manager accesses these logic-centric services via the *batcher* service sitting in the intermediary layer. This service acts as a façade over the algorithms and allows users to batch operations via simple scripts that can be submitted through the repository manager. For example, one could search for all models similar to an input process model, merge the result and visualize it in a given notation according to specific abstraction preferences.

The basic layer also hosts a set of data-centric services which serves as an interface to access the underlying persistent data – the core of the repository. Each data-centric service wraps one or more specific data entities and exposes the conventional functionalities of the related DBMS. These include data storage and retrieval, access control, integrity control and transactionality. Five data entities compose this layer:

- *models archive*: business process models in their original XML formats, e.g. BPMN models in XPDL (Workflow Management Coalition) or EPC models in EPML (Mendling et al.). These can be reference process models for specific domains, individualizations, single models or model collections. Moreover, these models can be configurable (e.g. a configurable reference process model);
- *canonical models archive*: a canonical representation of each of these models in XML. This canonical format filters out the specificities of a process modeling language, allowing the various repository algorithms to operate on a common process definition (more details in Section 5);
- *annotations archive*: metadata associated with the canonical models, e.g. layout information for visualization, or search indexes. This metadata is captured in the form of annotations to canonical models and organized in profiles;
- *patterns archive*: reusable libraries of process definitions for specific industry verticals, defined in canonical format. These can be used, e.g. for model creation, completion or pattern-based evaluation;
- *relations archive*: the relations between canonical representations of different process models, e.g. relations be-

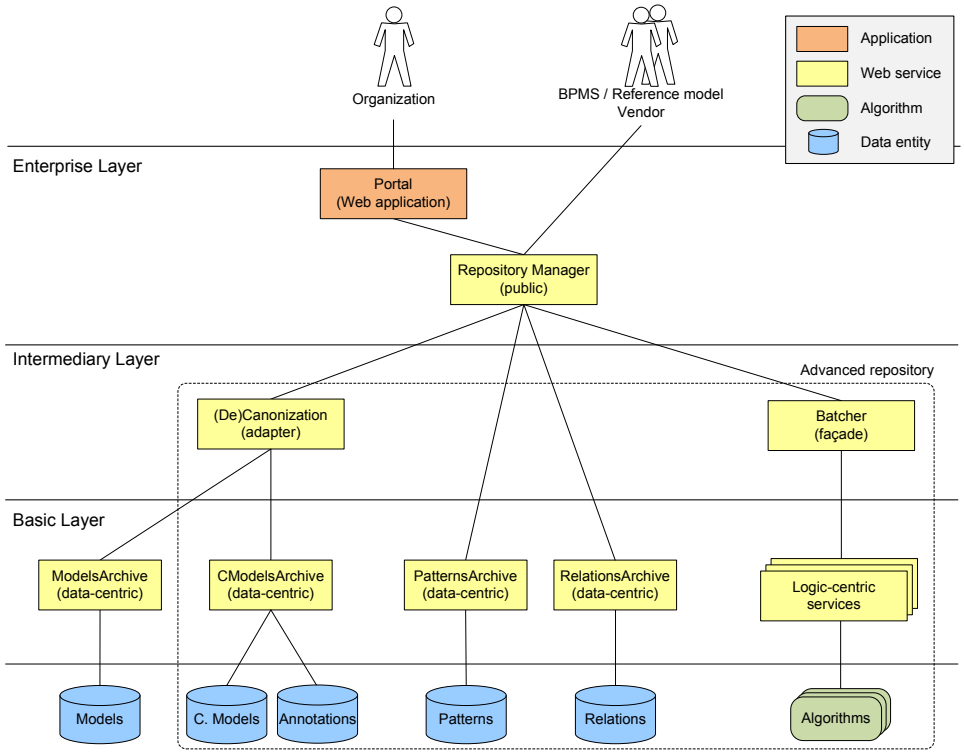

Figure 1: Service-oriented architecture of a process model repository.

tween configurable process models and their individualizations, or between process models and their extensions, used for change notification and adaptation control.

The repository manager accesses both process models and their canonical representation via the *(de)canonization* service – an intermediary adapter equipped with format conversion capabilities. This service is invoked the first time a request is made by the user to check-in a new process model in the repository. This service converts a copy of the model from its original format into its canonical representation; the latter model is indexed by the repository manager and stored in the canonical models archive. From that time onwards, a process model is always accessed through its canonical representation, although it is also persisted in the models archive in its original format (synchronization between the two formats is achieved through metadata). Since the algorithms operate on the canonical format, new models generated through these algorithms, e.g. when merging a set of similar processes into a configurable model, are produced directly in the canonical format. The (de)canonization service is also invoked to convert a canonical representation of a process into some tool or language-specific format when users check-out content from the repository, e.g. when importing a newly created model into a third-party application or into their file system.

5. Canonical Process Format

The canonical process format provides a common, unambiguous representation of business processes captured in different notations and/or at different abstraction levels, such that all process models can be treated alike. The idea behind this format is to represent only the structural characteristics of a process model that are common in the majority of modeling languages. Language-specific concepts are omitted because they cannot be meaningfully interpreted when dealing with process models originating from different notations, i.e. when "cross-language" operations are performed. Moreover, this canonical format is agnostic to graphical information such as shapes, line thickness and positions, which is contained in a concrete process definition. This information is stored separately in the form of annotations, and only used when a canonical model needs to be presented to the user or converted to an original format.

We identify five advantages in using a canonical format for the provision of advanced process model repository functionalities:

1 *Standardization*: a canonical format makes it possible to standardize software access to process definitions via a set of APIs. This is achieved through the (de)canonization adapter, which allows the various algorithms to work on a common process structure. In this way, cross-language operations can be directly performed and concatenated, i.e. without the need to first convert a model into another model's notation.

2 *Efficiency*: avoiding language conversions in turn improves the overall system efficiency. Moreover, annotations can be used to index canonical elements with specific meanings, with the purpose to expedite queries. In fact, searching large collections for models having particular properties may be very time consuming. Thus hav-

ing a single optimized format to avoid on-the-fly ad-hoc conversions is definitely preferable from a performance point of view.

3 *Interchangeability*: annotations also capture non-structural aspects of a process model, such as graphical information or process semantics, which can be automatically inferred from a concrete process definition. By organizing these annotations in profiles, a profile inferred from a process model can be applied to another canonical model, and a canonical model can have multiple profiles. In this way it is e.g. possible to switch between different graphical representations while keeping the same process structure.

4 *Reusability*: the canonical format is also used as the format for storing business process patterns and industry reference models. On the one hand, this facilitates the execution of those operations that involve such content, e.g. conformance analysis or pattern-based completion. On the other hand, it makes this content virtually available in every process modeling language that is supported by the repository.

5 *Flexibility*: the elements of a canonical format are defined through an inheritance mechanism such that at the highest abstraction level a process is simply seen as a directed, attributed graph. This allows algorithms to treat process models at different levels of granularity, depending on the type of operation required by the user.

We observe that without a common process format, a variant of each algorithm would need to be implemented for every (new) process modeling language. Moreover, conversions from one language to another would need to be put in place, to allow cross-language operations such as comparisons and merges.

In the next section we provide a detailed description of the canonical format adopted in APROMORE.

5.1. Metamodel

The metamodel of the canonical process format is defined using the UML class diagram shown in Figure 2. A CanonicalProcess is a container for a set of Nets, ResourceTypes and Objects. Each Net is a directed, attributed graph made up of Nodes and Edges, and represents a process or a subprocess. The top process is indicated as 'root', while all other Nets are marked as 'subnets'. Nodes can be of type Routing or Work, while Edges represent links between Nodes. Routing nodes capture all elements of a process model which are used for routing purposes (i.e. no work is performed from a business perspective), and as such they have more than one incoming edge and/or more than one outgoing edge. They can be Splits (ORSplit for inclusive data-driven choice, XORSplit for exclusive data-driven choice and ANDSplit for parallel branching), Joins (ORJoin for synchronizing merge, XORJoin for simple merge and ANDJoin for synchronization), and States (to indicate the state before an event-driven decision is made or soon after a merge). Splits have one incoming edge and multiple outgoing edges, Joins have multiple incoming edges and one outgoing edge, States can have multiple incoming and outgoing edges. The conditions upon which an (X)ORSplit choice is made, must be specified via the attribute 'condition' of each Edge leaving the (X)ORSplit. Also, one such an Edge can be marked as 'default' to indicate the default branch to be chosen if the conditions associated with all other Edges leaving the same Split evaluate to false.

Different from Routing nodes, Work nodes capture those elements of a process which are relevant from a business perspective. Work nodes have at most one incoming edge and one outgoing edge and can be partitioned into Tasks and Events. A Task node models a process element which actively performs some work as part of a process, e.g. preparing an invoice or processing a message. Task nodes can be atomic, or compound if they enclose a net describing their behavior. The enclosed net is indicated as 'subnet'. Events are used to signal the beginning or the end of a process, or to signal something that has happened during a process execution. Events can be specialized into Message events to capture a message being sent or receipt, and Time events to capture e.g. a timeout or a delay.

Work nodes can be associated with one or more ResourceTypes and Objects. Each ResouceType captures a class of organizational resources participating in the process, i.e. a group of concrete resources rather than the resources themselves. These can be Human, e.g. a position or role in an organization, or Nonhuman, e.g. an information system or equipment. For instance, the Human ResourceType "Finance Officer" may refer to the set of persons of an organization with role Finance Officer. ResourceTypes can have one or more specializations, e.g. "Finance Officer" may be specialized in "Senior Finance Officer" and "Junior Finance Officer". This relation is transitive and antisymmetric, and typically indicates a separation of duties. Each association between a Work node and a ResourceType indicates that a resource of that ResourceType is required to carry out the Work node. Therefore, a Work node associated with the same ResourceType n times, means that n resources of that ResourceType are required to carry out the given Work node (e.g. this captures the concept of teamwork for human resources, i.e. a set of persons all working on the same task). The association between Work nodes and ResourceTypes can specify a 'qualifier' to indicate the status a given ResourceType takes when performing the associated Work node, e.g. only one person of all the persons with role "Finance Officer" associated with Work node "Prepare invoice" is qualified as "Responsible" person. The association between Work node and ResourceType can be 'optional' to indicate that the work may be performed without involving the specific resource (see the attribute optional of resourceTypeRef).

Objects capture organizational business objects that are involved in the process. These can be physical artifacts, e.g. a paper-based invoice (Hard object) or information artifacts, e.g. a file or variable representing an electronic invoice (Soft object). For the latter, the 'type' of the object must be specified, e.g. the file extension or variable type. Objects can be associated with a Work node via an 'input' relation if they are utilized by the Work node, and/or via an 'output' relation if they

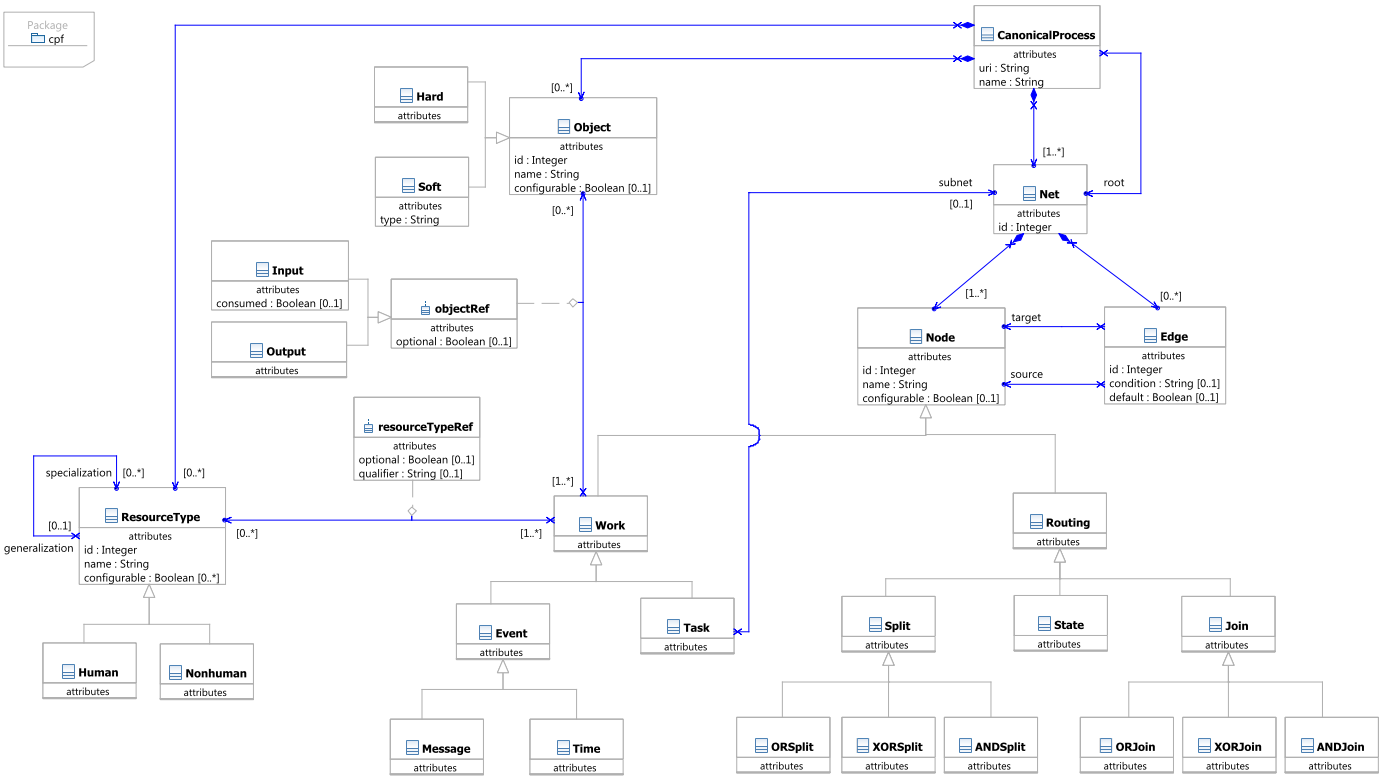

Figure 2: The UML metamodel of the canonical process format (association cardinalities of 1 are omitted).

are produced by the Work node. These relations correspond to read/write operations in the case of Soft objects. An object used as both input and output of a Work node indicates that the object is updated, e.g. an invoice is filled-out or a variable changes its content. Moreover, input objects can be marked as 'consumed' if they are destroyed while being used by a Work node. Similar to ResourceTypes, the association between Objects and Work nodes can also be tagged 'optional' to capture a situation where the Work node may be performed without using or producing the specific object.

Nodes, ResourceTypes and Objects can be configurable. This is denoted by their optional attribute 'configurable'. A node's configuration options are indicated through annotations outside a canonical representation. Configuration is an important aspect for large model repositories. However, given the diversity of languages and concepts, the configuration mechanism itself is not part of the canonical process format.

In the next section we motivate the choice for such elements and show how these are mapped to elements in concrete process modeling languages.

5.2. Methodology and Mapping

The elements to be included in the canonical format were identified from an analysis of commonalities among six widely adopted business process modeling languages. These are two languages for conceptual process modeling: EPCs (Keller et al., 1992) and BPMN 1.2 (Object Management Group, 2008), and four languages for executable process modeling: Protos 8.0,[3] WF-Nets (Aalst, 1998), YAWL 2.0 (Aalst and Hofstede, 2005) and WS-BPEL 2.0 (OASIS, 2007).

To date, EPCs (Event-driven Process Chains) are probably the most used process modeling notation among practitioners. Initially developed for the design of the SAP R/3 reference process model (Curran and Keller, 1997), they later became the core modeling language of the ARIS platform, and were adopted by other vendors for the design of SAP-independent reference models (e.g. the ARIS-based reference models for ITIL (IDS Scheer, 2009a) or SCOR (IDS Scheer, 2009b)). We support EPCs along with two extensions: eEPCs (extended EPCs) (Scheer, 1999) and iEPCs (integrated EPCs) (La Rosa et al., 2008). These cater for the representation of organizational resources and objects participating in a process, and for the representation of variation points on top of these elements respectively.

BPMN (Business Process Modeling Notation) is an emerging notation that can be seen as alternative to EPCs. BPMN was designed with the intent to enable both business users and technical developers to model readily understandable graphical representations of business processes. The BPMN specification is driven by the OMG (Object Management Group) standardization committee and is supported by a growing number of organizations and IT vendors.

WF-Nets (Workflow nets) are a class of Petri nets specif-

[3] `http://www.pallas-athena.com`

ically designed for modeling executable business processes (Aalst, 1998). As such, they benefit from a rich body of theoretical results, analysis techniques and tools (Murata, 1989). WF-Nets have been extensively applied in academia to the formal verification of business process models (Verbeek et al., 2001). Specifically, we chose to adopt the core WF-Net notation which does not feature triggers, explicit splits and joins, and hierarchical transitions, to avoid overlaps with the YAWL language.

Protos is the modeling language of Pallas Athena's BPM|One platform, which has been used for the design and implementation of various BPM solutions worldwide. Besides its use in practice, we chose Protos for the availability of a number of large models that the research team obtained via case studies conducted with European organizations.

YAWL (Yet Another Workflow Language) is an expressive language to describe, analyze and automate complex business process specifications, which builds on top of WF-Nets and provides comprehensive support for the Workflow Patterns (Aalst et al., 2003). YAWL is widely used in teaching and research, and is facing an increased uptake in industry (YAWL Foundation).

Finally, WS-BPEL (Web Service Business Process Execution Language) is an alternative to YAWL, which describes business process models as a composition of Web services. For this reason, BPEL represents a convergence between Web services and business process technology. Its specification derives from the joint effort of a number of IT vendors and has been standardized by the OASIS (OASIS, 2009) consortium.

We compared all modeling elements in the above languages with each other, by looking at the underlying concepts captured by each element. For example, EPC functions were compared with BPMN tasks, WF-Nets transitions, Protos activities, YAWL tasks and BPEL activities, since they all capture the concept of "performing some work". In order to do so, we first decomposed any concrete language construct in its fundamental concepts. For example, in YAWL splits and joins are always attached to tasks, so we extrapolated the join and split behavior from a YAWL task, and compared the former two with the routing elements of the other languages (more details on the conversion of language constructs are provided below). From this analysis, we created a canonical element for each concept that was shared by at least four languages out of the six taken into account.

Table 2 illustrates how the canonical elements from Figure 2 are mapped to concrete modeling elements. We can observe that basic concepts such as 'Edge', 'Task', 'Event' and 'ANDSplit/Join' are shared by all languages, others such as 'State', 'XORSplit/Join' and 'Object' are only shared by five languages, while more advanced concepts such as 'ORSplit/Join', 'ResourceType' and the specific Event types 'Message' and 'Timer', are only supported by four out of six languages. The table uses the term *sese* (single-entry single-exit) to refer to model elements with one incoming and one outgoing arc. These elements are treated different from nodes where a flow splits or multiple flows join (more details are provided below).

Table 2 also lists at the bottom those elements that are not supported by the canonical format. These elements refer to concepts that are not sufficiently represented in the six languages examined, such as error handling, cancelations or multiple instance tasks. Hence, supporting these concepts would have led to canonical elements being too language-specific. Moreover, these concepts are typically not interpreted by the various algorithms available in the literature. Nonetheless, the canonical format can be extended without varying its core structure, should new concepts be needed in future.

Figure 3 concludes the discussion on the canonical format by illustrating the canonical representation of thirteen common language constructs, according to the mapping in Table 2. Each construct (central column) is converted into a directed, attributed graph (right column) made up of a number of Nodes and Edges which are annotated with a reference to the respective element in the concrete language (trivial Edge annotations are omitted).

The first construct is taken from EPCs and represents a data-driven decision. EPCs only provide three modeling elements to capture a process control-flow: Functions, Connectors and Events. Functions are always mapped to Tasks while Connectors are mapped to Splits and Joins, depending on their type. Events are mapped to canonical Events if they do not immediately follow an (X)OR-split Connector, otherwise to Edges. This is because in EPCs the Events following an (X)OR-split are actually used to represent the conditions upon which the (X)ORSplit choice is made. Therefore, in this example the two EPC Events in the example of Figure 3 are mapped to two Edges, and their labels to the attribute 'condition' of these Edges.

The second and third constructs are two examples of syntactic sugar offered by BPMN, i.e. more concise representations of a given concept. The two incoming Flows to Task A in the second construct, are a compact notation for an Exclusive-join Gateway, while the two outgoing Flows are a compact notation for a Parallel-split Gateway. This translates to a canonical graph with one Work Node preceded by an XORJoin Node (mapping the implicit join) and followed by an ANDSplit Node (mapping the implicit split). Similarly, the third construct shows the compact notation for an Inclusive-split Gateway via two Conditional Flows (each capturing a condition for the choice) and one Default Flow (capturing the default condition). The corresponding canonical graph will have an ORSplit Node to capture the inclusive choice and one Edge for each outgoing Flow, with the Default Flow being mapped into an Edge with attribute 'default' set to true.

The fourth construct shows how AND-join and AND-split are modeled in WF-Nets and Protos. This behavior is never explicitly represented, but captured via incoming/outgoing Flows (or Connections in the case of Protos) to/from a Transition (called Activity in Protos). Furthermore, Protos allows one to specify the (X)OR-join and -split behavior for an Activity. Again, this is not explicitly shown and must be specified through an Activity's properties (fifth construct in Figure 3). In both cases, we need to add two extra Routing Nodes in the canonical representation to explicitly capture the split and join behavior, besides a Work Node to capture the Transition or Ac-

Canonical element	(e/i)EPC	BPMN 1.1	WF-Net	Protos 8.0	YAWL 2.0	WS-BPEL 2.0
Net	EPC, Compound Function	(Sub)Process		(Sub)Process	(Sub)Net	Process, Scope
Edge	Arc, Event subsequent to (X)OR-split Connector	Sequence Flow, Conditional Flow, Default Flow	Arc	Connection	Flow	Sequence, Link
Task	Function	Task	Transition	Activity	Task	Assign, Empty, ExtensionActivity
Event	Event not subsequent to (X)OR-split Connector	Plain Event, Start Rule Event	Input Place, Output Place, sese Place	sese Status	Input Condition, Output Condition, sese Condition	[mapped directly to specific types Message and Timer]
Message Event		Message Event		Message Trigger	WSInvoker Task	Invoke, Receive, Reply, onMessage in Pick
Timer Event		Timer Event		Timer Trigger	Timer Task	Wait, on Alarm in Pick
ANDSplit	AND-split Connector	Parallel-split Gateway, Task's Parallel-split	Transition's AND-split	Activity's AND-split	Task's AND-split	Flow (opening tag), Source Link
ORSplit	OR-split Connector	Inclusive-split Gateway, Task's Inclusive-split		Activity's OR-split	Task's OR-split	
XORSplit	XOR-split Connector	Data-based Exclusive-split Gateway		Activity's XOR-split	Task's XOR-split	If (opening tag), While (opening tag), RepeatUntil (opening tag)
ANDJoin	AND-join Connector	Parallel-join Gateway	Transition's AND-join	Activity's AND-join	Task's AND-join	Flow (closing tag), Target Link
ORJoin	OR-join Connector	Inclusive-join Gateway		Activity's OR-join	Task's OR-join	
XORJoin	XOR-join Connector	Exclusive-join Gateway, Task's Exclusive-join		Activity's XOR-join	Task's XOR-join	If (closing tag), While (closing tag), RepeatUntil (closing tag) Pick (closing tag)
State		Event-based Exclusive-split Gateway	non-sese (Input/Output) Place	non-sese Status	non-sese (Input/Output) Condition	Pick (opening tag)
ResourceType (Human/Nonhuman)	Org. Unit (eEPC), Position (eEPC), Supporting System (eEPC), Role (iEPC)	Pool, Lane		Role, Role Group, Responsible (resource), Application	Role, Org. Group, Position, Capability Custom Service	
Object (Hard/Soft)	Object (eEPC,iEPC)	Data Object		Document, Folder, Data Group, Data element	Task Variable	Variable, For and Until in Wait and onAlarm
	ProcessInterface, Person (eEPC)	Terminate, Complex Gateway, Events: Link, Error, Multiple, Compensation, Cancel, Signal; Events on Task boundary, Multiple Instance, Block repetition, Adhoc, Transaction, Message Flow, Exception Flow		Multiple, Buffer, Team, Batch	Cancelation, Multiple Instance, Participant	Exit, ForEach, Throw, Validate, Handlers, Correlations

Table 2: Conversion chart for the canonical format, including concrete elements not supported (sese = single-entry single-exit).

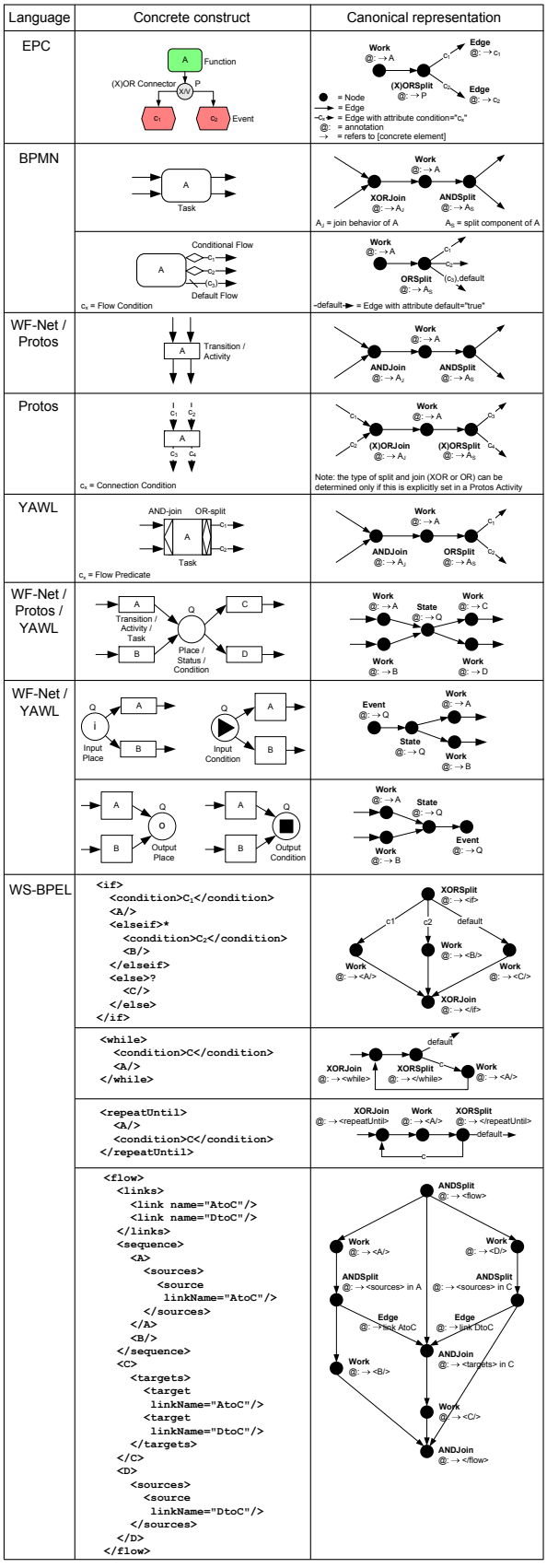

Figure 3: Canonical representation of common concrete language constructs.

tivity.

Unlike some of the other languages, in YAWL splits and joins must be explicitly represented as Task decorations (sixth construct in Figure 3). This is because they are semantically bound to the Task's behavior. Therefore in the canonical format we need to separate a Task from its routing behavior.

A Place in WF-Net, a Status in Protos and a Condition in YAWL, are all represented by a circle in the respective notations, and used to capture either a state of the process or an event occurring during the execution of a process. In the canonical format we separate these two concepts: if the Place (Status, Condition) is sese, we map it to an Event, whereas if it is non-sese, we map it to a State. This latter situation is shown by the seventh construct in Figure 3, where there are two incoming and two outgoing arcs.

A Place (Condition) is also used to capture the beginning/end of a WF-Net (YAWL) process. In this case we map the Place (Condition) to an Event irrespective of the number of outgoing/incoming arcs. However, if the Place (Condition) is non-sese, we also map it to a State. This can occur when an Input Place (Condition) has more than one outgoing arc or when an Output Place (Condition) has more than one incoming arc. In the first case, the added State is used to capture the state in which an event-driven decision is taken; in the second case, it is used to capture the final state before the process ends. The eight construct shows the canonical representation of a non-sese Input Place (Input Condition), while the ninth construct shows that of a non-sese Output Place (Output Condition).

The last four constructs show the canonical representation of the routing activities of WS-BPEL: If (to model exclusive data-driven choices), While and RepeatUntil (to model loops) and Flow (to model parallelism). The WS-BPEL standard does not define a standard visual representation for its processes, therefore we used the WS-BPEL XML format to illustrate these examples.

6. Prototype Implementation

As a proof of concept, we implemented a prototype of APROMORE according to the architecture described in Section 4.[4] The prototype supports the following features:

- *Basic functionalities*: model import/export, model search, model classification;
- *Comparison functionality*: similarity search;
- *Management functionality*: harmonization-driven creation.

These functionalities can be accessed via a Web portal, or directly by using the available Web services. The portal exposes the above functionalities through a graphical interface to provide process models visualization and editing capabilities (see Figure 4). Specifically, the portal is implemented using

[4]The propotype is available at `http://is.tm.tue.nl:8080/pros`

the so-called "model-view-controller" pattern, where the portal itself is merely a view on the models stored in the underlying database with Java methods acting as controllers. The algorithms for similarity search and harmonization-driven creation, as well as the (de)canonization adapter, are exposed as Web services through a standard WSDL interface.

Internally, both the models archive and the canonical models archive are implemented using a single MySQL database, although these are exposed as two separate logical entities through data-centric Web services. Currently, the process modeling languages that are supported are EPCs and BPMN.

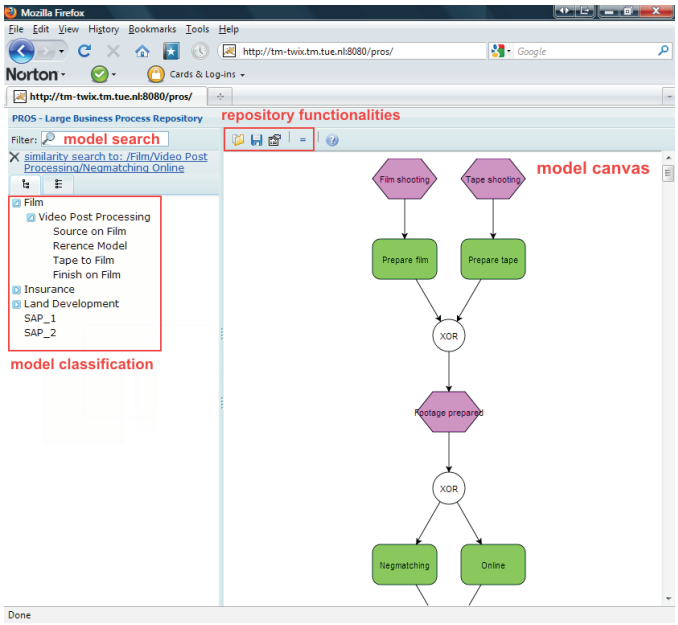

Figure 4: A screenshot of the prototype implementation of the Web portal.

In the following we describe two example scenarios that are supported by the prototype repository.

In the first scenario an organization can use the business process repository for advanced search functionalities. The collection of process models to be analyzed can originate from some proprietary BPMS and then imported into the repository. Depending on the functionalities provided by the external BPMS and the level of integration between the latter and the repository, BPMS users can profit from the advanced functionalities provided by the repository. For example, they can search for a particular model based on keywords, on models classification (e.g. per industry vertical) or by using an input model that is similar to the model to be searched for. A tighter integration is possible by invoking the Web services provided by the repository directly from the BPMS. Alternatively, a link is established between the repository Web-portal and the BPMS such that users can use the Web-portal to perform their searches and upon opening a process model in the portal, the BPMS is launched.

In the second scenario the harmonization-driven creation provided by the repository can be used to assist an organization integrating their process models with those of another organization, e.g. as part of a merger or acquisition. Both organizations can import their sets of business processes in the repository (provided that they are in a format that the repository supports). Subsequently, they can use the similarity search function to search for pairs of models that are similar. In a next step they can be aided in establishing a match between elements from one process and elements from the other process and merge the two models into a configurable process model, using this match. The resulting model will capture both the commonalities between the two models, and their differences, in the form of variation points. This new model can then provide a roadmap for implementing changes to the current business services and IT infrastructure supporting the business process, in order to rationalize them.

7. Conclusion

This paper presents APROMORE, which is an advanced repository to hold, analyze and re-use large sets of process models. APROMORE is an open source platform implemented according to the principles of service-oriented architectures, and exposed to the end user via a Web interface. The canonical format for process modeling notations is an essential ingredient for dealing with the diversity of available notations. A prototype demonstrating the feasibility of APROMORE is also presented in this paper.

The contribution of APROMORE can be discussed from two angles. Considering the interests of *practitioners*, the tool is thought to be helpful in dealing with many of the challenges that stakeholders face when dealing with large numbers of process models. In this respect, APROMORE goes well beyond the typical capabilities offered by commercial tools, e.g., basic access control and simple version control. APROMORE provides advanced support for dealing with models in different notations, platform-independent access, and maintaining the relations that exist between models. Also, APROMORE is capable of incorporating a collection of state-of-the-art techniques for analyzing, visualizing, transforming, and creating process models, which hitherto have mostly been known to the research community but hardly used in practice. In this way, APROMORE can be regarded to as a tangible means for knowledge transfer to the process modeling praxis.

From a *research* perspective, it is noteworthy that APROMORE is open to all researchers who have an interest in applying and developing techniques dealing with the analysis and optimization of process models. Due to its service-oriented architecture, it will be relatively easy for researchers to develop their own services and Web plug-ins, offering new capabilities while interacting with existing ones, by relying on the common infrastructure of APROMORE. In this way, APROMORE offers a separation of concerns that we hope is appreciated by our fellow researchers. An initiative that inspired us in this respect is ProM,[5] which is highly successful as an open research platform in the field of process mining.

[5] `www.processmining.org`

At present, one of the challenges is to populate the repository with large sets of process models to be used as a testbed and which can help to unleash the capabilities of APROMORE. Thanks to the involvement of four research groups in the APROMORE project, we have some 2000+ models available. Most of them are of medium size (20-100 tasks) and have been developed either in an academic or in an industrial context. At this stage, we are in touch with various industrial partners in the financial, healthcare, governmental, and creative industries domains to involve them in this initiative. Other challenges relate to more technical and operational issues, such as adding open, Web-based interfaces to proprietary implementations of analysis techniques, ensuring that the hardware can scale with the use in APROMORE, aligning with access models such as the OpenId initiative,[6] and integrating with other open platforms such as ProM and the Oryx visual editor[7].

With respect to future research, our efforts will be devoted to the development of new analysis and management techniques to be integrated in APROMORE. One example would be the development of an advanced version control system that can provide a semi-automatic resolution of conflicting process model updates. This is a relevant characteristic for a modern collaborative process modeling environment where it is realistic to assume that many stakeholders with different skills and responsibilities will partake in the modeling activity, thus potentially generating conflicting versions that need to be harmonized.

In conclusion, we believe that APROMORE is an important step in reaching a more mature level of dealing with the management of massive collections of process models. This entails challenges, but is highly relevant for practice, and serves as an exciting area for research. We hope that both practitioners and researchers will join us in the further development of APROMORE.

References

W.M.P. van der Aalst. Structural Characterizations of Sound Workflow Nets. Computing Science Reports 96/23, Eindhoven University of Technology, Eindhoven, 1996.

W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.

W.M.P. van der Aalst. TomTom for Business Process Management (TomTom4BPM). In *Proceedings of the 21st International Conference on Advanced Information Systems Engineering (CAiSE'09) – keynote*, pages 8–12, 2009.

W.M.P. van der Aalst and T. Basten. Identifying Commonalities and Differences in Object Life Cycles using Behavioral Inheritance. In J.M. Colom and M. Koutny, editors, *Application and Theory of Petri Nets 2001*, volume 2075 of *Lecture Notes in Computer Science*, pages 32–52. Springer, 2001.

W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245–275, 2005.

W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.

W.M.P. van der Aalst, M. Dumas, F. Gottschalk, A.H.M. ter Hofstede, M. La Rosa, and J. Mendling. Preserving Correctness During Business Process Model Configuration. *Formal Aspects of Computing*, 2009. (forthcoming).

W.M.P. van der Aalst, J. Nakatumba, A. Rozinat, and N. Russell. Business Process Simulation: How to get it right? In J. vom Brocke and M. Rosemann, editors, *International Handbook on Business Process Management*. Springer, 2010.

R. Anupindi, S. Chopra, S.D. Deshmukh, J.A. van Mieghem, and E. Zemel. *Managing Business Process Flows*. Prentice Hall, 1999.

A. Awad, G. Decker, and M. Weske. Efficient compliance checking using bpmn-q and temporal logic. In M. Dumas, M. Reichert, and M.-C. Shan, editors, *Proc. of the 6th International Conference on Business Process Management*, volume 5240 of *Lecture Notes in Computer Science*, pages 326–341. Springer, 2008. ISBN 978-3-540-85757-0.

S. Balko, A. Barros, A.H.M. ter Hofstede, M. La Rosa, and M. Adams. Controlled flexibility and lifecycle management of business processes through extensibility. In W. Esswein, J. Mendling, and S. Rinderle-Ma, editors, *Proceedings of the 4th Workshop on Enterprise Modelling and Information Systems Architectures (EMISA)*, volume 152 of *Lectures Notes in Informatics*, pages 97–110. GI, 2009. (forthcoming).

W. Bandara, G. Gable, and M. Rosemann. Factors and measures of business process modelling: model building through a multiple case study. *European Journal of Information Systems*, 14(4):347–360, 2005.

J. Becker, M. Rosemann, and C. von Uthmann. Guidelines of business process modeling. In W.M.P. van der Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management. Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 30–49. Springer, 2000.

P.A. Bernstein. Applying model management to classical meta data problems. In *Conference on Innovative Data Systems Research (CIDR)*, pages 209–220, 2003.

P.A. Bernstein and U. Dayal. An Overview of Repository Technology. In *Proceedings of the 20th Very Large Data Bases Conference (VLDB'94)*, pages 705–713. M. Kaufmann, 1994.

R.W. Blanning. Model management systems: an overview. *Decision Support Systems*, 9(1):9–18, 1993.

T. Curran and G. Keller. *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*. Upper Saddle River, 1997.

B Curtis, M.I. Kellner, and J. Over. Process modeling. *Communications of the ACM*, 35(9):75–90, 1992.

I. Davies, P. Green M. Rosemann, M. Indulska, and S. Gallo. How do practitioners use conceptual modeling in practice? *Data & Knowledge Engineering*, 58(3):358–380, 2006.

A. K. Alves de Medeiros, W.M.P. van der Aalst, and A.J.M.M. Weijters. Quantifying process equivalence based on observed behavior. *Data & Knowledge Engineering*, 64(1):55–74, 2008.

R.M. Dijkman. A Classification of Differences between Similar Business Processes. In *Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC'07)*, pages 37–50, Annapolis, Maryland, USA, 2007. IEEE.

R.M. Dijkman, M. Dumas, and L. Garcia-Banuelos. Graph matching algorithms for business process model similarity search. In *Proceedings of the 7th International Conference on Business Process Management (BPM'09)*, LNCS. Springer, 2009.

D.R. Dolk and B.R. Konsynski. Knowledge representation for model management systems. *IEEE Transactions on Software Engineering*, 10(6):619–627, 1984.

P. Effinger, M. Kaufmann, and M. Siebenhaller. An Interactive Layout Tool for BPMN. In *BPMN 2009 – 1st International Workshop on BPMN (CEC09 - 11th IEEE Conference on Commerce and Enterprise Computing)*, 2009.

R. Eshuis and P.W.P.J. Grefen. Constructing customized process views. *Data & Knowledge Engineering*, 64(2):419–438, 2008.

D. Fahland, C. Favre, B. Jobstmann, J. Koehler, N. Lohmann, H. Voelzer, and Karsten Wolf. Instantaneous soundness checking of industrial business process models. In *Proceedings of BPM 2009*, Lecture Notes in Computer Science. Springer-Verlag, 2009.

C. Fillies, G. Wood-Albrecht, and F. Weichhardt. Pragmatic applications of the Semantic Web using SemTalk. *Computer Networks*, 42(5):599–615, 2003.

G.M. Giaglis. A Taxonomy of Business Process Modeling and Information Systems Modeling Techniques. *International Journal of Flexible Manufacturing Systems*, 13(2):209–228, 2001.

F. Gottschalk, Wil M. P. van der Aalst, and M.H. Jansen-Vullers. Merging event-driven process chains. In R. Meersman and Z. Tari, editors, *Proceedings of the 16th International Conference on Cooperative Information*

[6] `http://openid.net`

[7] `http://bpt.hpi.uni-potsdam.de/Oryx`

Systems (CoopIS'08), volume 5331 of *Lecture Notes in Computer Science*, pages 418–426. Springer, 2008.

T.R.G. Green and M. Petre. Usability analysis of visual programming environments: A 'cognitive dimensions' framework. *J. Vis. Lang. Comput.*, 7(2): 131–174, 1996.

G. Grossmann, Y. Ren, M. Schrefl, and M. Stumptner. Behavior based integration of composite business processes. In W.M.P. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, editors, *Business Process Management, 3rd International Conference, BPM 2005, Nancy, France, September 5-8, 2005, Proceedings*, volume 3649 of *Lecture Notes in Computer Science*, pages 186–204. Springer, 2005.

J.A. Gulla and T. Brasethvik. On the Challenges of Business Modeling in Large-Scale Reengineering Projects. In *Proceedings of the 4th International Conference on Requirements Engineering*, pages 27–38. IEEE Computer Society Washington, DC, USA, 2000.

C.W. Günther and W.M.P. van der Aalst. Fuzzy mining – adaptive process simplification based on multi-perspective metrics. In G. Alonso, P. Dadam, and M. Rosemann, editors, *Proceedings of the 5th International Conference on Business Process Management (BPM'07)*, volume 4714 of *Lecture Notes in Computer Science*, pages 328–343. Springer, 2007.

K. van Hee, O. Oanea, N. Sidorova, and M. Voorhoeve. Verifying generalized soundness for workflow nets. In I. Virbitskaite and A. Voronkov, editors, *Proc. of the 6th International Conference on Perspectives of System Informatics, PSI'2006*, volume 4378 of *Lecture Notes in Computer Science*, pages 231–244, Novosibirsk, June 2006. Springer-Verlag.

M. Hepp, F. Leymann, C. Bussler, J. Domingue, A. Wahler, and D. Fensel. Semantic business process management: Using semantic web services for. business process management. In *IEEE International Conference on e-Business Engineering*, pages 535–540, Beijing, China, 2005. IEEE.

IDS Scheer. ARIS ITIL. Home Page, 2009a. `http://www.ids-scheer.com/en/ARIS/ARIS_Reference_Models_/ARIS_ITIL/3742.html`. Accessed: August 2009.

IDS Scheer. ARIS SCOR. Home Page, 2009b. `http://www.ids-scheer.com/en/ARIS/ARIS_Reference_Models_/SCOR/81882.html`. Accessed: August 2009.

M. Jarke, M.A. Jeusfeld, C. Quix, and P. Vassiliadis. Architecture and quality in data warehouses: An extended repository approach. *Information Systems*, 24(3):229–253, 1999.

D. Karagiannis and H. Kühn. Metamodelling Platforms. Invited Paper. In K. Bauknecht and A. Min Tjoa and G. Quirchmayer, editor, *Proceedings of the 3rd International Conference EC-Web 2002 - Dexa 2002, Aix-en-Provence, France*, volume 2455 of *Lecture Notes in Computer Science*, pages 182–196, 2002.

D. Karastoyanova, T. van Lessen, F. Leymann, Z. Ma, J. Nitzsche, B. Wetzstein, S. Bhiri, M. Hauswirth, and M. Zaremba. A reference architecture for semantic business process management systems. GITO-Verlag, Berlin, 2008.

G. Keller, M. Nüttgens, and A.-W. Scheer. Semantische Processmodellierung auf der Grundlage Ereignisgesteuerter Processketten (EPK). Veröffentlichungen des Instituts für Wirtschaftsinformatik, University of Saarland, Saarbrücken, Germany, 1992. (in German).

I. Kitzmann, C. König, D. Lübke, and L. Singer. A Simple Algorithm for Automatic Layout of BPMN Processes. In *BPMN 2009 – 1st International Workshop on BPMN (CEC09 - 11th IEEE Conference on Commerce and Enterprise Computing)*, 2009.

M. Klein and A. Bernstein. Towards high-precision service retrieval. *IEEE Internet Computing*, 8(1):30–36, 2004.

M. La Rosa. *Managing Variability in Process-Aware Information Systems*. Ph.d. thesis, Queensland University of Technology, Brisbane, Australia, April 2009.

M. La Rosa, M. Dumas, A.H.M. ter Hofstede, J. Mendling, and F. Gottschalk. Beyond Control-Flow: Extending Business Process Configuration to Roles and Objects. In Q. Li, S. Spaccapietra, E. Yu, and A. Olivé, editors, *Proceedings of the 27th International Conference on Conceptual Modeling (ER'08)*, volume 5231 of *Lecture Notes in Computer Science*, pages 199–215. Springer, 2008.

H. Lee and J.-W. Joung. An enterprise model repository: Architecture and system. *J. Database Manag.*, 11(1):16–28, 2000.

F. Leymann and W. Altenhuber. Managing business processes as an information resource. *IBM Systems Journal*, 33(2):326–348, 1994.

S. Melnik, E. Rahm, and P.A. Bernstein. Rondo: A programming platform for generic model management. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 193–204. ACM New York, NY, USA, 2003.

J. Mendling. *Metrics for Process Models: Empirical Foundations of Verification, Error Prediction and Guidelines for Correctness*, volume 6 of *Lecture Notes in Business Information Processing*. Springer, Berlin, Germany, 2008.

J. Mendling. Empirical studies in process model verification. *LNCS Transactions on Petri Nets and Other Models of Concurrency*, 2:208–224, 2009.

J. Mendling and C. Simon. Business Process Design by View Integration. In Johann Eder and Schahram Dustdar, editors, *Proceedings of BPM Workshops 2006*, volume 4103 of *Lecture Notes in Computer Science*, pages 55–64, Vienna, Austria, 2006. Springer-Verlag.

J. Mendling, M. Nüttgens, and M. La Rosa. EPML Schema 2.0. `http://www.processconfiguration.com/schemas/EPML_2.0.xsd`. Accessed: August 2009.

J. Mendling, H.A. Reijers, and J. Cardoso. What makes process models understandable? In Gustavo Alonso, Peter Dadam, and Michael Rosemann, editors, *Business Process Management - BPM 2007*, volume 4714 of *Lecture Notes in Computer Science*, pages 48–63. Springer, Brisbane, Australia, 2007.

J. Mendling, J. Recker, and H.A. Reijers. On the usage of labels and icons in business process modeling. *International Journal of Information System Modeling and Design*, 2009a. (forthcoming).

J. Mendling, H.A. Reijers, and W.M.P. van der Aalst. Seven process modeling guidelines (7pmg). *Information and Software Technology (IST)*, 2009b. In Press.

M. Momotko and K. Subieta. Process query language: A way to make workflow processes more flexible. In G. Gottlob, A.A. Benczúr, and J. Demetrovics, editors, *Advances in Databases and Information Systems, 8th East European Conference, ADBIS 2004, Budapest, Hungary, September 22-25, 2004, Proceesing*, volume 3255 of *Lecture Notes in Computer Science*, pages 306–321. Springer, 2004.

T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

OASIS. *Web Services Business Process Execution Language (WS-BPEL), Version 2.0. OASIS Standard*. OASIS, 2007. `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf`. Accessed: August 2009.

OASIS. Home Page. `http://www.oasis-open.org`. Accessed: August 2009, 2009.

Object Management Group. Home Page. `http://www.omg.org`. Accessed: August 2009.

Object Management Group. *Business Process Modeling Notation (BPMN), Version 1.1. OMG Specification*. OMG, 2008. `http://www.bpmn.org/Documents/BPMN\%201-1\%20Specification.pdf`. Accessed: August 2009.

A. Polyvyanyy, S. Smirnov 0002, and M. Weske. Process model abstraction: A slider approach. In *12th International IEEE Enterprise Distributed Object Computing Conference, ECOC 2008, 15-19 September 2008, Munich, Germany*, pages 325–331. IEEE Computer Society, 2008.

G. Preuner, S. Conrad, and M. Schrefl. View integration of behavior in object-oriented databases. *Data & Knowledge Engineering*, 36(2):153–183, 2001.

H.A. Reijers, R.S. Mans, and R.A. van der Toorn. Improved Model Management with Aggregated Business Process Models. *Data & Knowledge Engineering*, 68(2):221–243, 2009.

M. Rosemann and Wil van der Aalst. A Configurable Reference Modelling Language. *Information Systems*, 32:1–23, 2007.

A. Rozinat, M.T. Wynn, W.M.P. van der Aalst, A.H.M. ter Hofstede, and C.J. Fidge. Workflow simulation for operational decision support using design, historic and state information. In M. Dumas, M. Reichert, and M.-C. Shan, editors, *Business Process Management, 6th International Conference, BPM 2008, Milan, Italy, September 2-4, 2008. Proceedings*, volume 5240 of *Lecture Notes in Computer Science*, pages 196–211. Springer, 2008.

A.-W. Scheer. *ARIS – Business Process Frameworks*. Springer, 3rd edition, 1999.

A.-W. Scheer and M. Nüttgens. Aris architecture and reference models for business process management. In W.M.P. van der Aalst, J.Desel, and A. Oberweis, editors, *Business Process Management, Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 376–389. Springer, 2000.

M. Schrepfer, J. Wolf, J. Mendling, and H.A. Reijers. The impact of secondary notation on process model understanding. In *Proceedings of Practice of En-*

terprise Modeling (PoEM'09), Lecture Notes in Business Information Processing, 2009.

J. Siegeris and O. Grasl. Model driven business transformation – an experience report. In M. Dumas, M. Reichert, and M.-C. Shan, editors, *Proceedings of the 6th International Conference on Business Process Management (BPM'08)*, volume 5240 of *Lecture Notes in Computer Science*. Springer, 2008.

A. Streit, B. Pham, and R. Brown. Visualization support for managing large business process specifications. In W.M.P. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, editors, *Proceedings of the 3rd International Conference on Business Process Management (BPM'05)*, volume 3649 of *Lecture Notes in Computer Science*, pages 205–219. Springer, 2005.

T. Theling, J. Zwicker, P. Loos, and D. Vanderhaeghen. An architecture for collaborative scenarios applying a common bpmn-repository. In L. Kutvonen and N. Alonistioti, editors, *Distributed Applications and Interoperable Systems, 5th IFIP WG 6.1 International Conference, DAIS 2005, Athens, Greece, June 15-17, 2005, Proceedings*, volume 3543 of *Lecture Notes in Computer Science*, pages 169–180. Springer, 2005.

L.H. Thom, J.M. Lau, C. Iochpe, and J. Mendling. Extending business process modeling tools with workflow pattern reuse. In J. Cardoso, J. Cordeiro, and J. Filipe, editors, *ICEIS 2007 - Proceedings of the Ninth International Conference on Enterprise Information Systems, Volume EIS, Funchal, Madeira, Portugal, June 12-16, 2007*, pages 447–452, 2007.

B.F. van Dongen, R.M. Dijkman, and J. Mendling. Measuring similarity between business process models. In *Proceedings of the 20th International Conference on Advanced Information Systems Engineering (CAiSE)*, volume 5074 of *LNCS*, pages 450–464. Springer, 2008.

H.M.W. Verbeek, T. Basten, and W.M.P. van der Aalst. Diagnosing Workflow Processes using Woflan. *The Computer Journal*, 44(4):246–279, 2001.

C. Ware, H.C. Purchase, L. Colpoys, and M. McGill. Cognitive measurements of graph aesthetics. *Information Visualization*, 1(2):103–110, 2002.

I. Weber, I. Markovic, and C. Drumm. A conceptual framework for composition in business process management. *Lecture Notes in Computer Science*, 4439: 54, 2007.

Workflow Management Coalition. XPDL Home Page. `http://www.wfmc.org/xpdl.html`. Accessed: September 2009.

YAWL Foundation. YAWL Adoption. `http://www.yawlfoundation.org/about/adoption.html`. Accessed: August 2009.