Towards a Compendium of Process Technologies
The jBPT Library for Process Model Analysis

Artem Polyvyanyy[1] and Matthias Weidlich[2]

[1] Queensland University of Technology, Brisbane, Australia
artem.polyvyanyy@qut.edu.au
[2] Technion – Israel Institute of Technology, Haifa, Israel
weidlich@tx.technion.ac.il

Abstract. This paper presents the idea of a compendium of process technologies, i.e., a concise but comprehensive collection of techniques for process model analysis that support research on the design, execution, and evaluation of processes. The idea originated from observations on the evolution of process-related research disciplines. Based on these observations, we derive design goals for a compendium. Then, we present the jBPT library, which addresses these goals by means of an implementation of common analysis techniques in an open source codebase.

Keywords: Compendium, process technology, process analysis, jBPT

1 Introduction

Study of *processes*, i.e., orderings of observed events, is integral to the field of information systems. Usually, processes are studied by means of models. A *process model* is a particular representation of processes of the same nature. A *workflow model* and a *business process model* are examples of process models within the information systems discipline. Many research activities that deal with the design, execution, and evaluation of processes, rely on analysis of process models. Results of these endeavors are often validated by means of prototypical implementations. Such prototypes serve as a proof of concept and provide other researchers with a reference to enter competition for better results.

Unfortunately, such prototyping is usually hindered by different programming languages, frameworks, availability of other scientific prototypes and their reusability. Moreover, a considerable portion of a researcher's time must often be spent on making results applicable for different contexts, e.g., for process models captured in different modeling languages, such as BPMN and EPC. This holds even though these different contexts usually share similar characteristics. For instance, languages for specifying process models are rather similar w.r.t. the core modeling elements and their semantics.

This paper proposes *design principles* for a compendium of *process technologies* and discusses a concrete instantiation of these principles in the jBPT project[1]. The project develops a library of often used resources and algorithms when

[1] jBPT stands for "Business Process Technologies for Java".

dealing with analysis of process models. The library is implemented in Java and published[2] as open source under the GNU Lesser GPL license. The Java language is often chosen as a programming language for scientific prototypes due to its "architecture-neutral and portable" principle, which is also summarized in the Java promise of "write once, run anywhere" (despite of the fact that this claim is still arguably questionable). The jBPT project enjoys the cross-platform benefits of Java but also incorporates its own design philosophy which aims at maximal reuse of developed techniques.

The core design principle of the compendium is to bridge the heterogeneity of different process modeling languages by a unified framework that builds upon the most generic notions of graph theory. The methods and techniques included in the jBPT library are implemented for the common entities of different modeling languages and, thus, often can be reused for models captured in different languages. This principle has found its wide use in numerous research prototypes.

The next section lists observations on research that led to the idea of a compendium of process technologies. Sect. 3 summarizes design goals for the compendium. Sect. 4 presents the jBPT library (an instantiation of the compendium design), gives a usage example, outlines its functionality, and illustrates its maturity by enumerating research projects which rely on the library. Sect. 5 positions jBPT among other research frameworks, before we conclude the paper.

2 Observations on Process Research

In order to derive requirements for a compendium of process technologies, we start with a number of observations on process-related research.

The Nucleus of Process Modeling Languages. First and foremost, we observe that the plethora of process modeling languages, all defining their own syntax, semantics, and notation, largely hides the fact that most languages have a common conceptual root. On the one hand, they can essentially be seen as control flow graphs, with nodes relating to functional entities (what is done?) and edges defining the coordination for process execution (how is it done?). Clearly, process models capture more than the functional and the control flow perspective. However, most other perspectives, such as data handling and organizational structures, are often attached to the control flow graph (e.g., by partitioning the graph with swimlanes for organizational roles). Hence, the control flow graph can be seen as the backbone of any process model. On the other hand, we observe that control flow graphs are not arbitrary graphs, but are structured by 'common behaviors' of processes. Execution alternatives, concurrency, repetitive behavior, and hierarchical refinement are basic behavioral patterns that are supported by all major process modeling languages. Although languages define different mechanisms for realizing these patterns, they lead to similar control flow graphs.

The Need for Evaluation. In recent years, there has been a clear trend towards empirical evaluation of process research. Depending on the type of the concept that is to be validated, user studies or experiments with large

[2] `http://code.google.com/p/jbpt/`

(and often real-world) datasets are conducted. Consequently, virtually all newly proposed concepts and techniques are implemented at least in prototypes. Yet, most evaluations reported in research papers are not reproducible. This is only partly due to the non-availability of non-disclosed datasets. Often, the actual implementations are either not available or are hard to run and extend in the absence of a generic research framework for process model analysis. Also, we observe that a lot of functionality clones exist across these research prototypes, a result of redundant development efforts. Generic research frameworks may overcome these problems as successfully demonstrated in certain areas, e.g., by the ProM framework [1] for process mining research.

Broad Applicability of Structural or Behavioral Concepts. Another remarkable trend in process research is the broad applicability of certain concept and techniques for a large number of research questions. That is, a number of structural and behavioral concepts and techniques are on the one hand specific to process models (and thus not covered by frameworks for graph analysis), but on the other hand general enough to address a variety of research issues. Prominent examples are the Refined Process Structure Tree [2,3] and Behavioral Profiles [4] of a process model. The former has been applied, for instance, for modeling support, verification of process models, and clone detection in large repositories. The latter turned out to be useful for consistency analysis, the definition of re-usable action patterns, and compliance checking. A compendium of process technologies that features these techniques, therefore, supports the creation of prototypes in a broad context.

3 Design Goals

Based on the observations outlined in the previous section, this section enumerates design goals for a compendium of process technologies.

Reuse of Resources and Algorithms. The core idea behind the design of a compendium is reuse of functionality in all possible contexts. Resources and algorithms needed to conduct process model analysis should be applicable in different contexts. Here, different contexts are often dictated by different modeling languages. As detailed above, these languages usually operate within the common notion of a control flow graph. Therefore, the support of several abstraction levels when working with the notion of a control flow graph, which might range from a concrete syntax specification of a modeling notation to a generic graph-based formalism, is identified as a fundamental requirement for the design of a compendium. Every resource and every algorithm from a compendium should address the most abstract level of the control flow notion possible. This principle allows maximal reuse of techniques across all contexts that share the same abstraction level of concepts under consideration.

Don't Repeat Yourself. Every implementation of an analysis technique should be unique within a compendium. All tests and optimizations should target this unique implementation.

Effective Modularity. It is expected that a concrete realization of a compendium will comprise a large collection of methods and techniques. Therefore,

the availability of mechanisms for its effective modularization is essential. A compendium should allow natural identification of its independent functional components, which may be separated and/or recombined.

Utility Functionality. Research prototypes often require utility functionality besides the actual implementation of the analysis technique. An example would be the support of different serialization formats. In particular, serialization formats for common modeling notations and means to generate graph definitions used by standard software for graph visualization, e.g., `dot` graph description language [5], are required to support the implementation of analysis techniques.

Publishing Philosophy. To support researchers during the implementation of research prototypes, a compendium of process technologies needs to be publicly available with little restrictions on it usage. Further, extensions and future developments of a compendium are fostered by an open source publishing philosophy. In particular, a compendium published under the GNU Lesser GPL[3] ensures that any extensions will be available for the research community.

4 The jBPT Library

To achieve the design goals for a compendium of process technologies, we developed the jBPT library. In this section, we review the essentials of this library by first sketching its architecture and giving an illustrative example of how the library can facilitate development of process model analysis techniques. Finally, we lists some of the most used functionality of the library and give an overview of its application in concrete research projects.

Architecture. An excerpt of the core structure of jBPT is shown in Fig. 1. It illustrates the interface and class hierarchy to capture different types of graphs, starting with multi-hypergraphs as the most general model. It also shows that all graph models are typed with generics, which allows, for instance, to define a `PetriNet` as an `AbstractDirectedGraph` with parameter `E` being bound to `Flow` classes and parameter `V` to `Node` classes. A `Flow` class represents a specialization of a directed edge, i.e., a `AbstractDirectedEdge` with `V` being bound to `Node` classes. The definition of process model classes, or more specific classes for BPMN or EPC models, and Petri nets as specializations of different kinds of graphs is the basis for extensive reuse of analysis algorithms. Those may be implemented on the according level of the graph hierarchy and then be used for all specializations.

Illustrative Example. To illustrate how jBPT facilitates the development of research prototypes, we consider two properties of process models. First, *structural soundness*[4] relates to the structure of a process model, see [7], and holds if the model has at least one source node (no direct predecessors) and at least one sink node (no direct successors), such that each node of the model lies on a path from some source to some sink. Intuitively, execution is instantiated at some of the source nodes and terminates eventually at some of the model's sink nodes. Moreover, each node should have a chance to contribute to some

[3] All changed versions of a GNU Lesser GPL program must be released as free software.
[4] Not to be confused with the classical notion of soundness, a behavioral criterion [6].

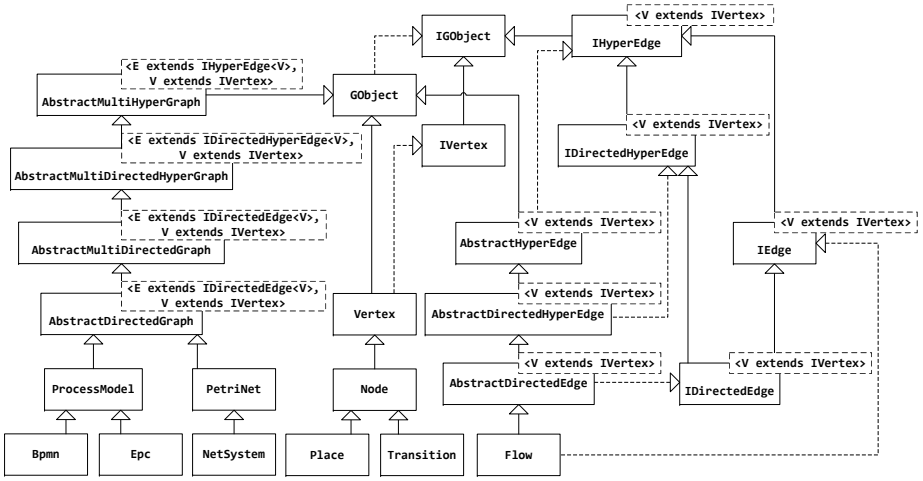

Fig. 1. Excerpt of the class and interface hierarchy of jBPT

execution of the model. Second, we consider testing a Petri net for the structure of a Workflow net [6], i.e., whether it has a single source node, a single sink node, such that every node is on some directed path from the source to the sink. Clearly, the first question is more generic and relates to process models in general. The latter, Workflow net structure, strengthens the notion of structural soundness and is typically applied to Petri nets only.

Checking for structural soundness is equivalent to a check of *strong connectedness*[5] of an enhanced version of the directed graph which is used to describe a process model, see *one possible implementation* in the `isMultiTerminal` function in `DGA.java` listing below. The *enhanced* graph is obtained by introducing a fresh source vertex (`src`) and a fresh sink vertex (`snk`), lines 8–10, to the graph, and then adding a fresh edge from the fresh sink to the fresh source. A directed graph describes a structurally correct process model if its enhanced version is strongly connected. A directed graph is strongly connected if it constitutes a single *strongly connected component* (SCC)[6]. Strongly connected components of directed graphs are discovered in $O(|V|+|E|)$ time with V and E as the sets of vertices and edges of the graph [8]. Hence, the `isMultiTerminal` check of a graph is done in linear time to its size. Note that prior to returning a result, we remove vertices `src` and `snk` (line 14), which implies removal of all adjacent edges. The `isTwoTerminal` function (lines 18–23) restricts the `isMultiTerminal` check to a single source and a single sink. Functions `isMultiTerminal` and `isTwoTerminal` can be applied to every Java object which implements the `IDirectedGraph` interface instantiated with `IDirectedEdge` and `IVertex` types, e.g., a `ProcessModel`, an `EPC`, or a `PetriNet`.

[5] A directed graph is called strongly connected if there is a directed path from each vertex in the graph to every other vertex.

[6] A strongly connected component of a directed graph is its maximal strongly connected subgraph.

As such, a method to implement the structural soundness check for an `EPC`, simply requires calling the respective method.

DGA.java – Collection of algorithms to manipulate directed graphs.

```
1 : public class DGA<E extends IDirectedEdge<V>,V extends IVertex> {
2 :    ...
3 :    public boolean isMultiTerminal(IDirectedGraph<E,V> g) {
4 :       if (g==null) return false;
5 :       Collection<V> sources = this.getSources(g);
6 :       Collection<V> sinks = this.getSinks(g);
7 :       if (sources.isEmpty() || sinks.isEmpty()) return false;
8 :       V src = g.getFreshVertex(); V snk = g.getFreshVertex();
9 :       for (V v : sources) g.addEdge(src,v);
10:       for (V v : sinks) g.addEdge(v,snk);
11:       g.addEdge(snk,src);
12:       SCCs<E,V> sccs = new SCCs<E,V>();
13:       boolean result = sccs.isStronglyConnected(g);
14:       g.removeVertex(src); g.removeVertex(snk);
15:       return result;
16:    }
17:
18:    public boolean isTwoTerminal(IDirectedGraph<E,V> g) {
19:       if (g==null) return false;
20:       if (this.getSources(g).size()!=1 || this.getSinks(g).size()!=1)
21:          return false;
22:       return this.isMultiTerminal(g);
23:    }
24: }
```

Below, the generic usage of the outlined functionality is illustrated by the Workflow net check for `PetriNet` objects. Function `isWorkflowNet` simply wraps the `isTwoTerminal` function for `PetriNet` objects, see lines 5–6 below.

PetriNetStructuralClassChecks.java – Structural tests for Petri nets.

```
1 : public class PetriNetStructuralClassChecks {
2 :    ...
3 :    public static boolean isWorkflowNet(PetriNet net) {
4 :       if (net==null) return false;
5 :       DGA<Flow,Node> dga = new DGA<Flow,Node>();
6 :       return dga.isTwoTerminal(net);
7 :    }
8 : }
```

jBPT Functionality. Besides already mentioned object models for various process modeling languages, the jBPT library offers a collection of supporting management routines, such as parsing and serialization to standard formats. Further, the functionality of the jBPT library comprises, among others, basic graph algorithms, connectivity-based decompositions of graphs and related process model related applications, e.g., the RPST [3], models and algorithms for behavioral profiles [9], algorithms for process model similarity (based on the

graph edit distance or behavioral relations), techniques from the theory of net systems and net systems unfolding, transformations of process models, etc. For an up-to-date list of all resources and functionality included in the library, please refer to the jBPT page at Google Code (`http://code.google.com/p/jbpt/`).

Applications. The idea of a compendium of process technologies as implemented by jBPT proved indeed valuable for the implementation of various research prototypes. Among others, the following projects are based on jBPT.

- `bpstruct`[7] is a collection of techniques proposed in [10] for transforming unstructured process models into well-structured ones.
- `BPM Academic Initiative`[8] is an academic version of a professional business process management tool. It integrates `bpstruct` through a REST API.
- `apromore`[9] is an open source repository to store and disclose process models of a variety of languages, such as BPMN, EPCs, YAWL, Workflow nets, etc. The jBPT initiative provides core functionality for the `apromore` platform. It underpins the canonical process model format of `apromore` and many of its core features and functionalities.
- `Flexab` [11] is a tool that allows for flexible abstraction of process models. It relies on algorithms for behavioral profiles as implemented in jBPT.

5 Related Work

There are several other research frameworks that have been designed in the spirit of a compendium of process technologies, yet with a different focus. A prominent example is the ProM framework [1] for process mining. It supports the creation of research prototypes with a plugin infrastructure that allows for implementing functionality based on the utilities provided by the ProM framework. In contrast to jBPT, the ProM framework does not focus on providing a set of common structural and behavioral analysis techniques, but rather features an infrastructure that also addresses the visualization of process models and includes a GUI for conducting experiments. Another platform for developing research prototypes for process management is the Oryx Editor [12]. It features a process model repository and a process model editor that are extensible in terms of modeling languages and the integration of analysis functionality. However, the Oryx Editor offers only limited support for implementing the actual analysis. Hence, it may serve as a front-end for the visualization and integration of functionality implemented in jBPT, as demonstrated by Flexab [11] for process model abstraction.

Various process model analysis techniques are available on demand in the Business Process Services[10] portal developed by IBM Research. These services can be invoked for a process model description and, for instance, conduct control flow analysis or refactor the model. The analysis of interactions between processes is at the core of the ServiceTechnology toolset[11], see also [13]. These tools

[7] `http://code.google.com/p/bpstruct/`
[8] `http://academic.signavio.com/`
[9] `http://code.google.com/p/apromore/`
[10] `http://business-services.researchlabs.ibm.com`
[11] `http://www.service-technology.org/`

implement techniques for service composition, discovery, and substitution. Both toolsets support the development of research prototypes by providing specific, encapsulated functionality and, thus, have a different focus compared to jBPT, which aims at supporting the actual development of prototypes with basic data structures and algorithms.

6 Conclusion

In this paper, we argued for the creation of a compendium of process technologies. The idea is to have a comprehensive collection of techniques for process model analysis, thereby supporting the implementation of research prototypes. Based on observations on process research, we derived design goals for a compendium. Further, we presented the jBPT library as a first implementation of that idea. This library offers a broad range of basis analysis and utility functionality and, due to its open publishing model, can easily be extended. It regularly attracts over 250 unique Internet visitors per month from more than 50 countries.

Acknowledgments. We highly appreciate the contributions to jBPT made by the community as detailed at `http://code.google.com/p/jbpt/people/list`.

References

1. Verbeek, H.M.W., Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: XES, XESame, and ProM 6. In: CAiSE Forum. Vol. 72 of LNBIP., Springer (2010) 60–75
2. Vanhatalo, J., Völzer, H., Koehler, J.: The refined process structure tree. Data Knowl. Eng. **68**(9) (2009) 793–818
3. Polyvyanyy, A., Vanhatalo, J., Völzer, H.: Simplified computation and generalization of the refined process structure tree. In: WS-FM. Vol. 6551 of LNCS., Springer (2010) 25–41
4. Weidlich, M., Mendling, J., Weske, M.: Efficient consistency measurement based on behavioral profiles of process models. IEEE Trans. Software Eng. **37**(3) (2011) 410–429
5. Gansner, E.R., North, S.C.: An open graph visualization system and its applications to software engineering. Software. Practice and Experience **30**(11) (2000) 1203–1233
6. van der Aalst, W.M.P.: Verification of Workflow nets. In: ICATPN. Vol. 1248 of LNCS., Springer (1997) 407–426
7. Weske, M.: Business Process Management: Concepts, Languages, Architectures, Second Edition. Springer (2012)
8. Tarjan, R.E.: Depth-first search and linear graph algorithms. SIAMCOMP **1**(2) (1972) 146–160
9. Weidlich, M.: Behavioural Profiles: A Relational Approach to Behaviour Consistency. PhD thesis, University of Potsdam (2011)
10. Polyvyanyy, A.: Structuring Process Models. PhD thesis, University of Potsdam (2012)
11. Weidlich, M., Smirnov, S., Wiggert, C., Weske, M.: Flexab — flexible business process model abstraction. In: CAiSE Forum. Vol. 734 of CEUR Workshop Proceedings., CEUR-WS.org (2011) 17–24
12. Decker, G., Overdick, H., Weske, M.: Oryx — an open modeling platform for the BPM community. In: BPM. Vol. 5240 of LNCS., Springer (2008) 382–385
13. Lohmann, N., Wolf, K.: How to implement a theory of correctness in the area of business processes and services. In: BPM. Vol. 6336 of LNCS., Springer (2010)