Towards true Process Descriptions Interoperability

Tomislav Rozman, Romana Vajde Horvat, Gregor Polančič

Institute of Informatics
Faculty of electrical engineering and computer science, University of Maribor
Smetanova 14, Maribor, SI-2000, Slovenia
Phone: (+386) 2 235 51 07 Fax: (+386) 2 235 51 34
E-mail: {tomi.rozman | romana.vajde | gregor.polancic}@uni-mb.si

Abstract. *The article describes a part of our research in the area of (business) process modeling, analysis and execution. It describes the current state of process languages and standards. It concentrates on the process model interoperability and portability problems, which are caused because of huge number of process modeling standards, which often do not even target the same conceptual level of process modeling. It provides an idea how to bring together those standards, using process concepts mapping into Petri Nets and incorporating the usage of process patterns. Transformation to strictly formal process languages has a potential to prove itself as a fair solution with the following advantages: the number of mappings among different process languages is reduced and the process model can be analyzed using proven Petri Net or general graph algorithms. The development of the approach is still in early phase. Mapping rules to/from process patterns are not strictly defined, neither trivial, therefore, they will be the main focus of our future research.*

Keywords. Process modeling, process analysis, transformation, Petri nets, BPM

1. Introduction

Business process engineer is often confronted with the dilemma, which 'complete, all-in-one process management solution' to choose, that fits his/her organization best. After a research (crawling through the jungle of a marketing materials and flashy presentations), he finds out, there are some nearly complete, but specific solutions. In fact, some of this so called 'business process management suites' are often legacy products, wrapped in new disguise. On the other hand, there is a great number of partial solutions. Those solutions are often intended for different audience; therefore they often do not share the same conceptual level. Some of these solutions are intended only for high-level graphical process modeling, other only for machine-level process model interchange and execution. Almost the same situation can be noticed for standard process modeling languages. It does not matter which solution is chosen, the result is, the process models created in specific environment are vendor dependent and cannot be simply plugged in different vendor's process management suite.

Some years ago, there were only a few standards for workflow description. Most promising one was developed by WfMC (WorkFlow Management Coalition) as the result of the incompatible workflow products of that time, but none of them had not really fully supported it. Never standards, based on XML, are promising, but their wider adoption is hindered due to great number of such languages [10].

Current development and research in the BPM area is focused on the separating the process definition from its execution environment. This approach presumes there is only one universal process description language, which can be interpreted by many process environments. But, at the time of writing of this paper, there are several competing standards, for example, BPML (Business Process Modeling Language), BPEL (Business Process Execution Language), XPDL (XML Process Definition Language), ebXML (Electronic Business XML Initiative).

Nowadays, a lot of researchers and organizations invent their own languages, standards or extensions to existing process modeling languages. There are several mainstream solutions for process description, which are under heavy evolution process.

Based on experiences from other research areas, which are dealing with the mapping of the real world 'things' to conceptual models, it is not probable that all-satisfying process modeling language will emerge.

Therefore, the transformations among different process modeling languages must exist, to keep compatibility among them. Because of great number of process description languages, a lot of mapping rules should exist also.

Therefore, we propose the solution, which reduces the number of mappings among different process modeling languages.

2. Paper organization

The rest of the paper is organized as follows: first, related work in this research area is summarized. Next, the basic idea of proposed approach is presented, which encompass the rationale for the research and briefly describes the proposed architecture. Further, the core elements of the architecture (process patterns mapping modules and workflow patterns) are explained. Possible applications of the proposed architecture are discussed.
Last chapters end with some future research directions and ideas.

3. Related work

Huge number of today's partially compatible process description languages increase the portability problems among them, as reported by many authors [7], [9], [10].

An effort to make a common process language is held by [10], that results in the development of the PSL (Process Specification Language), which has evolved from the PIF (Process Interchange Format) and KIF (Knowledge Interchange Format). PSL is also in the process of standardization, as a ISO/IEC 18629 standard [12]. PSL ontology is based on situation-calculus and consists of the following basic elements: activity, activity-occurrence, time-point and object, which are also the reason, why some authors [8] argue about suitability of PSL as the common process language. They propose Petri nets and pi-calculus as basics for the foundation language, mainly because today's process languages are based on message passing mechanisms and not on the time constraints. To make the situation more confusing, it is possible to transform PSL based process model into Petri nets [11].

An effort to make Petri nets more useful for workflow modeling is held by [3]. Research group of Department of Technology Management, Eindhoven University of Technology made a big contribution to characterization and analysis of modern, XML-based business process modeling languages, using workflow patterns. Static structure of those patterns can be described using WF-nets, which were introduced by [3].

The study of several mappings among process description languages was made, using specifications [13], [14], [15] and others.

4. Research method

At this stage, the non-reactive method was selected, because the proposed approach is under development and the idea is still evolving. The existing approaches are studied, compared and combined.

5. Basic idea

As was mentioned earlier, there is a problem about the number of mappings among different process modeling languages. If we want full interoperability among all the process description languages, there are *n(n-1)* mappings [9] in fully connected graph [Figure 1], where the nodes PLx represent process language, the arcs represent two-way mappings and the *n* is the number of nodes.

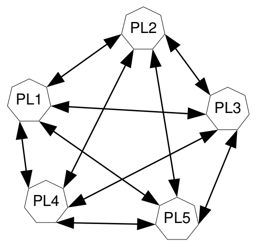


Figure 1 : The graph of mappings among process modeling languages

If the languages are XML based, the mapping among them becomes easier because of the possibility of the XSLT transformations usage, but the number of them stays the same. The problems arise, if the process description languages are not directly comparable, for example, if they cover different process modeling aspects. Therefore, some conditions are needed when transforming the process description from (PL1) to (PL2):

- Between the PL1 and the PL2 there must be a 'path', or, sequence of the defined mapping rules.
- The process concepts from every process description language on this path must be covered in its predecessor.

Fortunately that is the worst-case scenario.

Ideally, only one common ('core') process modeling language should exist, encompassing all of the possible process modeling concepts of all languages. In this ideal situation, the number of mappings is only *2*n,* where *n* represents the number of process languages. All mappings among process description languages are indirect; to transform the process description from language (PL1) to (PL2), the intermediate transformation to the 'core' language (CPL) is needed.

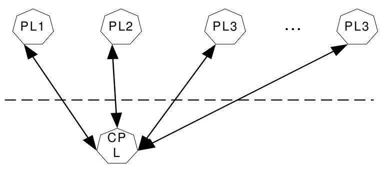


Figure 2 : Graph of mappings (one common process description language)

To realize this idea, the following conditions should be satisfied:

- The core language for the process descriptions should exist, with strong mathematical foundations and
- Two-way mappings to convert to and from the core process language should be defined.

In reality, some mainstream process modeling languages are evolving. The graph of mappings is also not fully connected, nor bi-directional, which means, only small set of mappings exists today. For example:

- BPMN (Business Process Modeling Notation) → BPML/BPEL [13],[14],
- XRL (eXtensible Routing Language) → PNML (Petri Net Markup Language).

Those process description languages, 'convergence points' or, de-facto standards reduce the number of mappings needed. The number of mappings here is between the positive and the negative scenario.

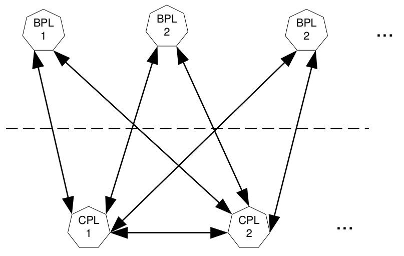


Figure 3 : Leveled graph of mappings among process modeling languages

Such topology is 'right' only if the core process description languages are compatible with one another, or they are derived from common low-level process language.

Relying on the current state of the development results on this field, majority of the parts of the proposed architecture and the stack of mappings [Figure 4] already exist. What is missing, is the model, mappings and relations, which connects high-level and low-level process description languages and formally defines mapping rules.

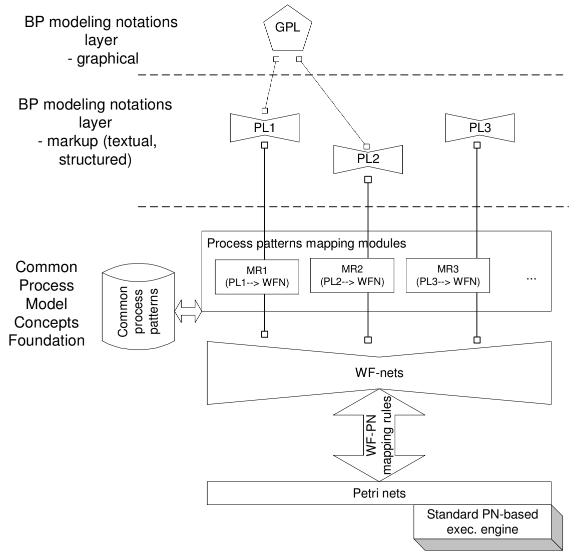

BP modeling notations layer - graphical
GPL
BP modeling notations layer - markup (textual, structured)
PL1
PL2
PL3
Process patterns mapping modules
MR1 (PL1--> WFN)
MR2 (PL2--> WFN)
MR3 (PL3--> WFN)
...
Common Process Model Concepts Foundation
Common process patterns
WF-nets
WF-PN mapping rules
Petri nets
Standard PN-based exec. engine

Figure 4 : Proposed mapping architecture

The proposed architecture contains the following tiers:

- The top-level tier, which consists of current and future process description languages, with only limited set of defined mappings among one another and
- Common Process Model Concepts Foundation layer, which consists of WF-nets and Petri net layer. On top of the WF-net layer is the most important part, the mapping layer, which converts variety

of process description languages to WF-nets.

For the core process description language, the Petri nets were chosen, because its advantages for a workflow modeling were proven many times.

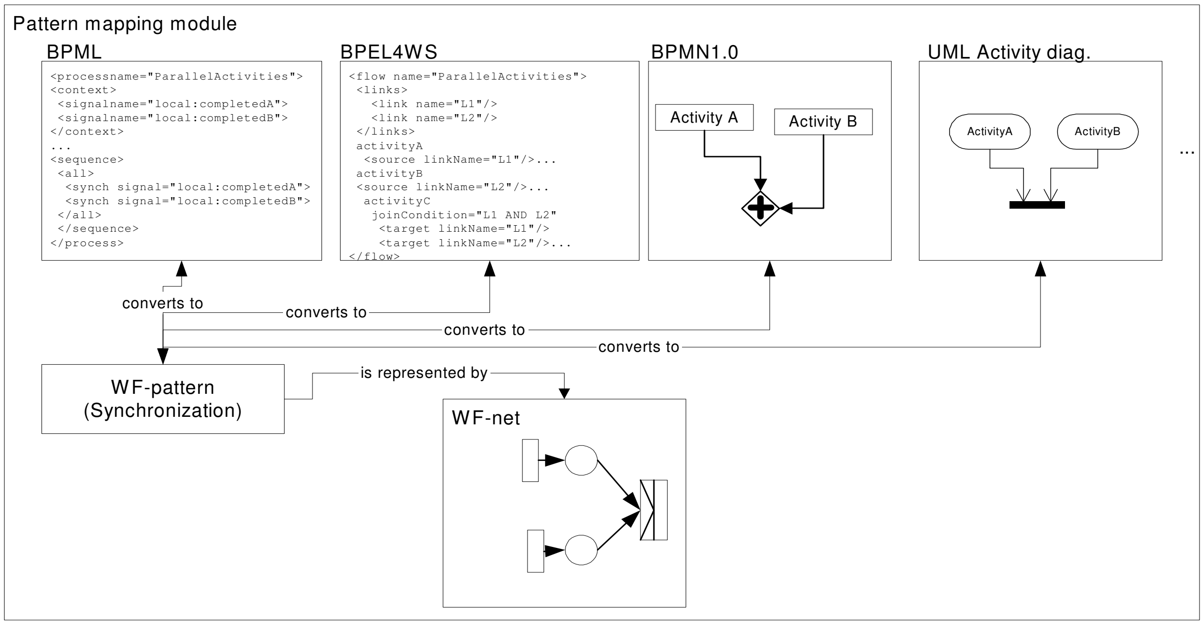
WF-nets were chosen, because an extensive analysis showed they are appropriate for process and workflow representation [3]. Main reasons are:

- WF-nets are high-level Petri nets, and they can be converted to basic Petri nets,
- WF-nets are more appropriate for a workflow modeling as basic Petri nets because they overcome their limitations (advanced routing and triggering constructs),
- WF-nets are Turing-complete, similar as other high-level Petri-nets, which means, any algorithm can be expressed and,
- WF-nets are very appropriate for formal notation of workflow patterns.

The core of the process patterns mapping modules [Figure 4] is the WF-pattern with several mapping rules defined.

6. Structure of the pattern mapping modules

The kernel of the pattern-mapping module [Figure 5] is the WF-pattern. Every WF-pattern can be represented with the WF-net, which can be transformed into Petri net. The mapping module contains mapping rules, for conversion to the specific process description languages.



```
<processname="ParallelActivities">
<context>
 <signalname="local:completedA">
 <signalname="local:completedB">
</context>
...
<sequence>
 <all>
  <synch signal="local:completedA">
  <synch signal="local:completedB">
 </all>
 </sequence>
</process>
```

```
<flow name="ParallelActivities">
 <links>
   <link name="L1"/>
   <link name="L2"/>
 </links>
 activityA
  <source linkName="L1"/>...
 activityB
 <source linkName="L2"/>...
  activityC
   joinCondition="L1 AND L2"
    <target linkName="L1"/>
    <target linkName="L2"/>...
</flow>
```

Figure 5 : Example of mapping rule for Synchronization pattern

The pattern-mapping module is extensible with additional mapping rules. Example of such pattern-mapping module is presented on [Figure 5]. The example pattern mapping module describes, how the pattern *synchronization* is presented in several process description languages: graphical (BPMN and UML Activity diagrams) and one markup language (BPML). The synchronization pattern represents a point in the process where multiple parallel branches converge into one single thread of control.

Examples of the mappings from *synchronization* to *BPML* and other WF-patterns are described in details in [4]. Other mappings are described in [5] (BPEL), [6] (XPDL), as a result of the pattern-based analysis of the web services composition languages.

WF-nets describe only static structure of the pattern-mapping module. For the definition of the WF-net semantics, different approaches are studied, for example, the PSL [11].

WF-nets and Petri nets does not have a mechanism to describe information objects, which are very important in a process modeling. Therefore, the usage of RDF (Resource Definition Framework) for the detailed token description is studied [9].

Because of the importance of WF-patterns in our approach, the following paragraph briefly describes them.

7. WF-patterns

The pattern is the abstraction from a concrete form, which keeps recurring, in a specific nonarbitrary contexts [2]. WF-pattern (Workflow pattern) is a combination of interconnected process modeling concepts, which formulate routing constructs, different synchronization or communication mechanism.

In the paper [3], the following groups of WF-patterns are identified:

1. *Basic control flow patterns*, which are the basic constructs present in most workflow and process description languages. They are used for sequential, parallel and conditional routing (Sequence, Parallel Split, Synchronization, Exclusive Choice, Simple Merge).
2. *Advanced branching and synchronization patterns* describe advanced types of splitting and joining behavior (Multi – choice, Synchronizing Merge, Multi – merge, Discriminator).
3. *Structural patterns are block structures* define workflow entry and exit points. In graphical languages, block structures are often considered to be too restrictive. Therefore, the patterns were identified, that allow a less rigid structure (Arbitrary Cycles, Implicit Termination).
4. *Patterns involving multiple instances* describe parts of the process, which needs to be instantiated several times (Multiple Instances Without Synchronization, Multiple Instances With a Priori Design Time Knowledge, Multiple Instances With a Priori Runtime Knowledge, Multiple Instances Without a Priori Runtime Knowledge).
5. *State-based patterns* describe state-based behavior of the process or workflow (Deferred Choice, Interleaved Parallel Routing, Milestone).
6. *Cancellation patterns* describe, how the process is canceled, based on incoming events and what are the following steps after cancellation (Cancel Activity, Cancel Case).

WF-patterns are conceptual constructs and they are not well covered in majority of process definition languages. This could lead to mapping problems and the specific process information can be lost. The following paragraph discusses possible work-around solutions.

8. The specific process model enhancements problem

Unfortunately, some process constructs are language specific and cannot be mapped to another process description language. On the other hand, some workflow patterns do not have direct representation in currently available process description languages. Sometimes, workaround solutions can be used. For example [4], BPML does not support *Multichoice*, *Discriminator* or *Synchronization merge* pattern directly, although this pattern can be expressed using combination of simpler concepts.

If we want true process descriptions interoperability, all vendor specific process information should be preserved. This information should be hidden and marked 'inactive', if the process model is mapped to the language, which does not support it.

Similar approach is used in some HTML code-generation tools. They create generic code and additional specific information, which is hidden in comments and are ignored by web browsers. Another example of information hiding is <head> part of HTML code. The <head> part is not rendered by the web browser (except for the <title></title> element), although it contains essential information for different use cases (web spiders and search engines).

9. Possible applications

Practical usage of presented approach is could be a web-based service of a process model repository and a translator. User, who wanted to transform his process model, would 'feed' the service with the process model and desired destination format. The service would check the model, transform it and return it to the user.

10. Discussion

Benefits of such architecture are, the process definition (model) could be independent not only from runtime (or, simulation environment), but also from the process language.

The main result of this approach is: (a) it will not matter any more in which process language the process is described and (b) the process models could be analyzed using proven Petri net or graph algorithms.

The idea is partly borrowed from the programming language evolution: first, there were hardware specific programming languages, then operating system specific languages and lastly, they developed into the problem area specific languages. Last ones can run on different platforms, because the source code is not compiled directly into a hardware and OS specific binary code, but into an intermediate level byte code, which is interpreted by virtual machine. Main programming language vendors adopt this approach: Sun with Java and Microsoft with C#.

It was recognized [1], that the closer the syntax, rules, and mnemonics of the programming language could be to "natural language" the less likely it became that the programmer would inadvertently introduce into the program. Similar thinking could be applied to process modeling languages.

11. Conclusion

We do not claim the proposed approach is exhaustive or optimal. It is a hybrid of multiple existing efforts, which try to create the universal process description and interchange language. It tries to take the advantages out of them.

At this stage, the proposed approach is at a draft stage.

Our future work will concentrate on further development of suggested approach: the mapping modules will be defined in details, refined and tested on practical, real-world examples.

12. References

[1] Evolution of High-Level Languages, http://www.encyclopedia.com/html/section/progrlan_EvolutionofHigh-LevelLanguages.asp [10/20/2003]

[2] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software. Professional Computing Series. Addison Wesley, Reading, MA, USA, 1995.

[3] Wil M.P. van der Aalst et al, Workflow Patterns: On the Expressive Power of (Petri-net-based) Workflow Languages, http://tmitwww.tm.tue.nl/research/patterns/download/invited_talk_cpn_2002.pdf [01/10/2004]

[4] Wil M.P. van der Aalst et al, Pattern Based Analysis of BPML (and WSCI), http://tmitwww.tm.tue.nl/research/patterns/download/qut_bpml_rep.pdf [01/10/2004]

[5] Wil M.P. van der Aalst et al, Pattern Based Analysis of BPEL4WS, http://tmitwww.tm.tue.nl/research/patterns/download/qut_bpel_rep.pdf [01/10/2004]

[6] Wil M.P. van der Aalst, Patterns and XPDL: A Critical Evaluation of the XML Process Definition Language, http://tmitwww.tm.tue.nl/research/patterns/download/ce-xpdl.pdf [01/10/2004]

[7] BPM & BIM, http://www.cbop.gr.jp/dl/semi/bosc/2001/011102bosc_05.pdf [03/01/2004]

[8] John F. Sowa, The WonderWeb Foundational Ontology Library and the DOLCE ontology, http://suo.ieee.org/email/msg08943.html [13/01/2004]

[9] Joshua Lubell, XML Representation of Process Descriptions, http://ats.nist.gov/psl/xml/process-descriptions.html [3/12/2003]

[10] Sharon Boyes- Schiller, Putting Workflow and BPM standards into context, e- Science Workflow Services, Edinburgh, [03/12/2003], http://www.nesc.ac.uk/talks/303/EdinburghPresentation2122003.pdf [02/02/2004]

[11] PSL home page, http://www.mel.nist.gov/psl/index.html [02/02/2004]

[12] ISO TC184/SC4, Setting the Standards for Industrial Data homepage, http://www.tc184-sc4.org/ [02/02/2004]

[13] BPMI.org, Business Process Modeling Notation, Working Draft (0.9), November 13, 2002, http://www.bpmi.org/bpmn-spec.esp [20/12/2002]

[14] BPMI.org, Business Process Modeling Notation, (1.0), http://www.bpmi.org/bpmn-spec.esp [25/09/2003]

[15] Assaf Arkin, Intalio, Business Process Modeling Language, http://www.bpmi.org/bpml-spec.esp [22/07/2002]