Managing Process Model Complexity - Part I: Concrete Syntax

Marcello La Rosa[1], Arthur H.M. ter Hofstede[1], and Petia Wohed[2]

[1] Queensland University of Technology, Australia
{m.larosa, a.terhofstede}@qut.edu.au
[2] Stockholm University/KTH, Sweden
petia@dsv.su.se

Abstract. While Business Process Management (BPM) is an established discipline, the increased adoption of BPM technology in recent years has introduced new challenges. One challenge concerns dealing with process model complexity in order to improve the understanding of a process model by stakeholders and process analysts. Features for dealing with this complexity can be classified in two categories: 1) those that are solely concerned with the appearance of the model, and 2) those that in essence change the structure of the model. In this paper we focus on the former category and present a collection of patterns that generalize and conceptualize various existing features. The paper concludes with a detailed analysis of the degree of support of a number of state-of-the-art languages and language implementations for these patterns.

Key words: Process model, pattern, complexity, presentation

1 Introduction

Business Process Management (BPM) deals with the life-cycle of business process models which includes their design, execution and analysis [39]. Through the application of BPM technology, businesses may realize cost reductions, time savings, and an increased agility to deal with change. The uptake and further development of this technology has seen a surge in recent years. Despite advancements in the field of BPM, both in academia and in industry, there are still challenges remaining that need to be dealt with in order to increase its uptake and its scope of application.

One of these challenges concerns the management of complex process models. Business process models may contain many elements which may have numerous and intricate control-flow dependencies between them. The more complex a business process model is, the harder it is to use it in the communication with the stakeholders (e.g. to determine if it properly reflects their business practices), to reason about it (both at design time and at runtime), and to evolve it over time (due to unforeseen circumstances or due to changing business practices).

There is a substantial body of literature on process model complexity and understandability on the one hand (e.g. [9, 21, 25, 28, 5, 35]) as well as on proposed mechanisms to deal with managing this complexity on the other hand (e.g. [37, 38, 23, 3]). However, these mechanisms are often too language-specific or not properly motivated. What is lacking is a clear overview of, and a motivation for, the various features that exist to managing complexity in process models. Such an overview could ultimately pave the way for more comprehensive support for complexity management in process modeling languages, standards and tools.

In this paper we follow the established approach of capturing a comprehensive range of desired capabilities through a collection of patterns (e.g. the well-known workflow patterns [2] provide a high-level, language-independent description of control-flow requirements in workflow languages). The patterns presented in this paper capture features to manage process model complexity as they are found in the literature, in process modeling languages or in their implementation. In order to scope the work, focus is exclusively on complexity reduction features that affect the appearance of a process model not its structure. Or, to put it differently, the patterns are concerned with aspects of concrete syntax not abstract syntax of a process model. The description of the patterns is language-independent, but typically their realization in one or more existing approaches is used to reinforce understanding and to demonstrate relevance. We conclude the paper with an overview of the degree of support for the patterns in a number of well-known process modeling languages and language implementations, thus providing an insight into comparative strengths and weaknesses.

2 Background

Syntactically speaking, a process model is a connected directed graph which contains a number of nodes and edges connecting them. The set of nodes can be partitioned into a number of subsets. One well established partition used by the Workflow Nets notation [4], consists of *transitions* and *places*. Another partition, used in eEPCs [19, 11], consists of *events*, *functions* and *connectors*. A third partition, applied in BPMN [30], consists of *activities*, *gateways*, and *events*. Common for all process notations is that they have a node type that captures the concept of *task* (in Workflow nets tasks are called transitions, in eEPCs functions, in BPMN activities/tasks, etc).

Process models are almost always graphically represented. The symbols used to represent the various types of nodes in the model are part of the *concrete syntax* of the process modeling language.The *abstract syntax* of a process modeling language is not concerned with representational aspects but deals with capturing the various types of process elements and the relationships between them. Hence, changing the graphical appearance of a process model (e.g. by rearranging nodes or modifying the iconic representation of a certain node type) should not have any consequences for its abstract syntax representation. In this paper focus is exclusively on representational matters, hence on concrete syntax.

In order to illustrate the various patterns, the BPMN (Business Process Modeling Notation) standard is used throughout this paper. An overview of the graphical representation of the main concepts of this notation can be found in Fig. 1. A detailed knowledge of this standard is not required to understand the various examples in this paper. For more information, the reader is referred to [30].

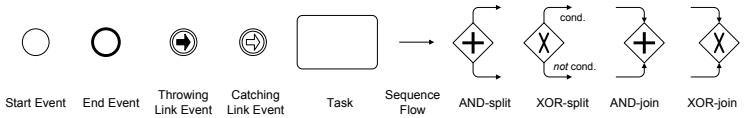

Fig. 1. BPMN 1.2 modeling elements used in this paper.

3 Patterns for Concrete Syntax Modification

We identified eight patterns operating on the concrete syntax of a process model and classified them according to the hierarchy in Fig. 2. Two layout patterns, *Layout Guidance* and *Layout Split*, describe features to modify the process model layout. Four highlight patterns outline visual mechanisms to emphasize certain aspects or parts of a process model. These are *Group Highlight*, *Graphical Highlight*, and two annotation patterns: *Pictorial Annotation* and *Textual Annotation*. Finally, *Explicit Representation* refers to the availability of explicit visual representations for a process construct, while *Naming Guidance* refers to naming conventions or advice to be used in a process model.

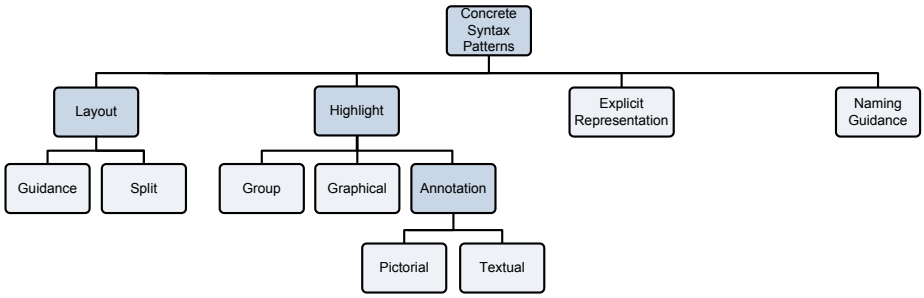

Fig. 2. Patterns for concrete syntax modification.

In the following we provide a detailed description of each pattern including the purpose, the rationale for its use (where available), its occurrence in languages, tools and in the literature, an example of its realization in BPMN, and the criteria used to evaluate how languages and tools support the pattern.

Pattern 1 (*Layout Guidance*)

Description This pattern refers to the availability of layout conventions or advice to organize the various model elements on a canvas. These include indications on the distribution (alignment) and orientation of modeling elements in the space.

Purpose To reduce clutter, especially in large process models or models that have undergone a number of updates.

Rationale Neat and tidy process models are generally easier to comprehend than chaotic and cluttered ones [25]. Crossing edges negatively affect model understanding [33].

Realization Some languages provide general guidelines on how a model should be laid out on the canvas. eEPCs prescribe to model processes from top to bottom [19], while the BPMN specification recommends "to pick a direction of Sequence Flow, either left-to-right or top-to-bottom" as well as to "direct the Message Flow at a 90° angle to the Sequence Flow" [30], p.30. Tool-wise, we can distinguish three categories. Some tools such as IDS Sheer ARIS offer sophisticated algorithms to layout both eEPCs and BPMN models, where elements orientation, alignment and spacing can be customized. Other, such as Oracle JDeveloper impose a fixed layout. A third category which includes the YAWL Editor, Sparx Enterprise Architect (EA) and Pallas Athena Protos, provides little or no layout support. In the literature, the problem of finding an optimal placement of model elements on the canvas has received quite some attention. Alpfelbacher et al. [6] suggest to place related elements spatially close to each other, Huotari et al. [17] and Purchase [33] point out that crossing edges should be avoided if possible, while Jensen [18] suggests that incoming and outgoing edges are placed on the opposite sides of a Colored Petri Net node to improve readability. BPMN-specific layout algorithms have been discussed in [20], while [14] provides a prototype implementation of a BPMN-Layouter tool. Finally, initial work towards determining the influence of various layout factors on process model understanding has been done in [35].

Example Fig. 3a shows a BPMN model that does not follow any layout guideline: i) the elements are not oriented in a consistent direction (e.g. the first two tasks have a top-to-bottom orientation, while the remaining ones are oriented from left-to-right); ii) subsequent elements are not closely positioned to each other (e.g. task Create new entry is far from task Insert invoice details and from the AND-split in-between); iii) there are several crossing edges. Fig. 3b shows the same model after repositioning the elements according to the BPMN guidelines.

Evaluation Criteria A language achieves full support if it provides layout conventions or advice. An implementation achieves full support if it provides layout algorithms that either follow the language layout conventions/advice or add proprietary extensions.

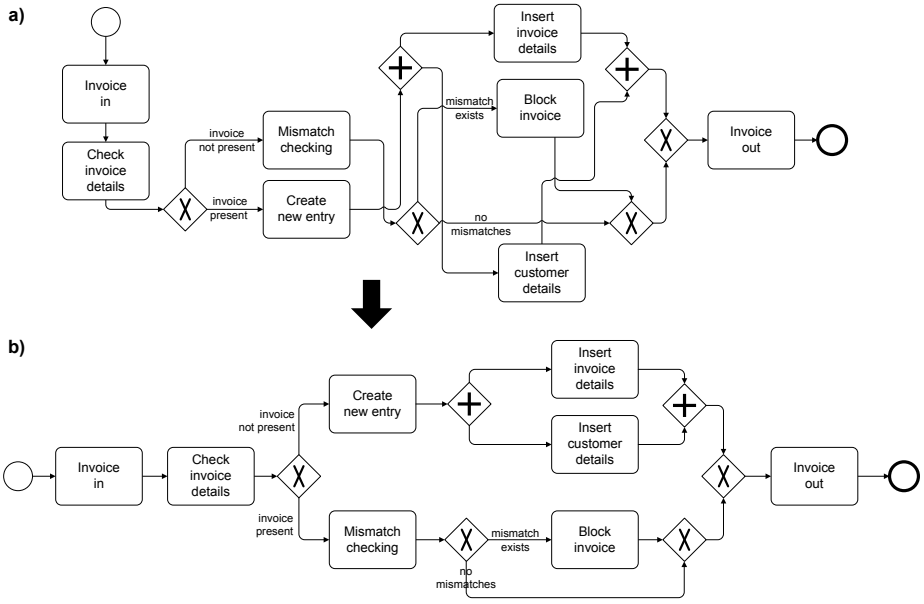

Fig. 3. a) A BPMN model not following any layout guidelines. b) The same model after applying the BPMN layout guidelines.

Pattern 2 (*Layout Split*)

Description This pattern refers to the availability of modeling constructs to divide a model in different parts. Such a division may be a logical model partition or purely necessitated by physical constraints, such as the size of the modeling canvas or of the printing page.

Purpose To split large models into several pages for presentation purposes. To reduce clutter in those models where crossing edges cannot be avoided.

Rationale Reducing model size positively affects model understanding [25].

Realization Some languages provide features to split a model in multiple parts. For example, BPMN offers the Link Event to allow the flow between two Flow Object elements to be graphically discontinued for space reasons, i.e. interrupted after a source Flow Object and resumed before a target Flow Object. The corresponding construct in UML Activity Diagrams (ADs) is the Activity Edge Connector while in eEPCs it is the ProcessInterface. In the literature, Effinger et al. [13, 14] define a mechanism to split large BPMN models. The idea is to cut those edges with numerous bends and crossings and insert two pointers at the two ends of each cut edge. The objective is to obtain subgraphs of nearly equal size while keeping the number of cut edges as low as possible.

Example The model in Fig. 3b captures the Invoice Processing fragment of a larger, end-to-end order-to-cash process model, where Invoice Processing in preceded by Ordering and followed by Delivery. This large model can thus be

represented over multiple pages, e.g. one page for each logical part. Fig. 4 focuses on the Invoice Processing part, connected to the other parts via BPMN Link Events. Moreover, the Invoice Registration part inside Invoice Processing has been modeled separately (logical separation). The essential thing is that all the model parts translate to the same conceptual structure since the splitting is applied to the model's concrete syntax.

Evaluation Criteria A language achieves full support if it has a construct that satisfies the pattern description. An implementation achieves full support if, in addition to the auxiliary language constructs, it allows navigating between the linked model parts. It rates as partial support if it allows the specification of splitting constructs only.

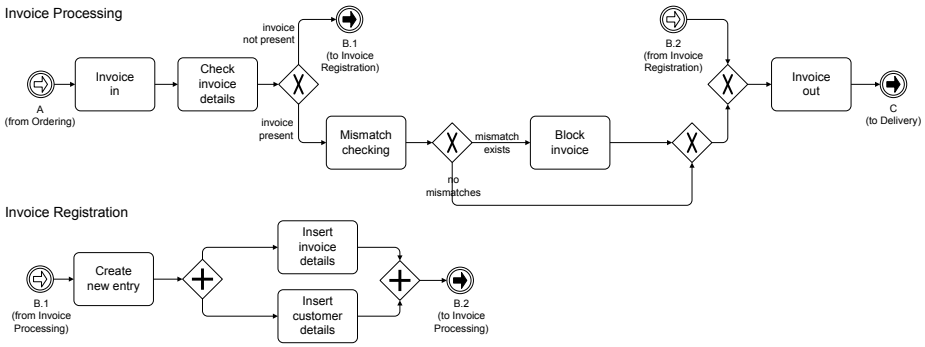

Fig. 4. An example of the use of Layout Split on the model in Fig. 3b.

Pattern 3 (*Group Highlight*)

Description This pattern refers to the availability of modeling constructs to visually group a set of logically-related model elements, and add a comment to characterize the group.

Purpose To visually accentuate a set of model elements based on some shared property, e.g. grouping all elements that need revision or all elements that refer to a given resource.

Realization BPMN is the only language that supports this pattern via the notion of Grouping – a dashed-line, rounded corner rectangle with a name. The elements in a BPMN Grouping are only grouped informally, without changing the model semantics. The majority of modeling editors provide a drawing palette to allow drawing a shape to group model elements, and to attach textual comments to the drawing. For example, ARIS allows one to draw shapes such as rectangles or circles, add a comment via the Free-form text feature, and group the shape with the text in one element. Similarly, in Protos a modeling area can be encircled via rectangles or ellipses. The background color of this area can be changed and a text area can be added to the model to leave comments. EA provides a non-UML element called System Boundary to define conceptual boundaries from a visual perspective.

Example Fig 5 shows the use of the BPMN Grouping construct to emphasize all tasks related to the SAP System and all tasks that need revision, for the model in Fig 3b.

Evaluation Criteria A language achieves full support if it has a construct that satisfies the pattern description. An implementation achieves full support if it has features that either follow from the language or add proprietary extensions.

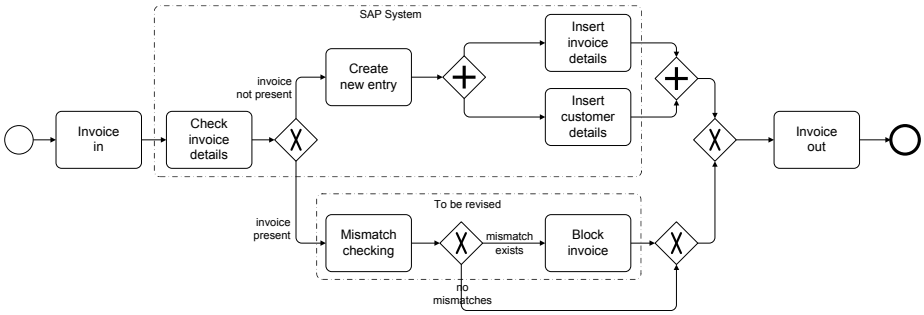

Fig. 5. An example of Group Highlight using the BPMN Grouping construct.

Pattern 4 (*Graphical Highlight*)

Description This pattern refers to the availability of features to change the visual appearance of model elements, such as shape, line thickness and type, background color, font type and color.

Purpose To visually accentuate some properties or aspects of model elements.

Rationale Appearance properties such as colors can reduce the cognitive overhead of associating syntactic elements with their semantics [22].

Realization eEPCs prescribe the use of different colors for each construct, e.g. *Functions* are represented in green, *Events* in purple, *Connectors* in grey and *Positions* in yellow. In Protos only the *Status* construct is colored in blue. BPMN allows flexibility in elements' size, color and line style, except for specific elements such as throwing and catching events, for which specific guidelines are indicated. The majority of modeling editors provide features to change the appearance of model elements. Those that support eEPCs such as MS Visio, ARIS and Oryx, visualize eEPC models in their default colors. In ARIS an element's background color, line thickness and line type can be changed, while in EA fonts' color can also be changed. Other tools such as Oryx and the YAWL Editor only allow customizing the background color. In the literature, the use of colors is suggested to identify edge ends and matching splits and joins in Workflow Nets [32], while in [12] the idea of color-coding matching splits and joins is implemented for the WoPeD tool. Moreover, in [13] a method is presented to represent different types of BPMN elements by objects differing in color and shape; in [16] color variations and line brightness are used to highlight the most significant behavior of unstructured process models mined from logs, while in [1] line thickness is suggested to indicate the most traversed process path.

Example Fig 6 uses colors to highlight matching splits and joins, and thick edges to highlight the most traversed path, for the model in Fig 3b.

Evaluation Criteria A language achieves full support if it provides guidelines to graphically emphasize certain elements. An implementation achieves full support if it follows the language guidelines and provides features to customize the visual appearance of modeling elements. It rates as partial support if the appearance of model elements cannot be customized.

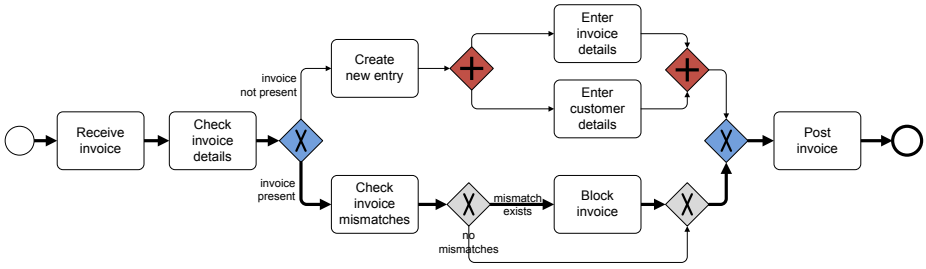

Fig. 6. Two examples of Graphical Highlight: coloring and line thickening.

Pattern 5 (*Pictorial Annotation*)

Description This pattern denotes the availability of features to assign pictorial elements, such as icons or images, to modeling elements, without changing semantics.

Purpose To strengthen model-specific concepts (e.g. annotating a receive task with an envelope), or to add domain-specific information (e.g. annotating a task with an exclamation mark to indicate criticality).

Rationale Associating pictorial elements with textual descriptions improves model understanding [31].

Realization In BPMN 2.0 [29], a task can be annotated with an icon indicating its type. For example, an empty envelope can be used to indicate a *Receive* task, while a hand can be used to indicate a *Manual* task. Similarly, Protos makes use of icons to distinguish among various activity types, e.g. Basic, Logistics, Authorize. Features to assign icons or images to modeling elements are recurrent in modeling editors. In some tools such as JDeveloper and Intalio|Designer icons are automatically assigned to tasks and cannot be customized. For example, in Intalio|Designer they are used to distinguish manual from automated BPMN tasks. In other tools, such as EA and the YAWL Editor, icons or images are fully customizable. For example, in EA one can replace the background of a UML activity with an image. In the literature, Mendling et al. [24] recognize the importance of annotating process models with icons to convey domain-specific information, and propose a set of 25 icons to graphically represent 25 frequently occurring task label categories.

Example In Fig 7 each task from Fig 3b has been annotated with an icon. For example, task "Block Invoice" features an icon indicating danger (explicative purpose) while task "Check invoice details" features a lens (reinforcing purpose).

Evaluation Criteria A language achieves full support if it has a construct that satisfies the pattern description. An implementation achieves full support if it offers features to customize icons/images attached to modeling elements. It rates as partial support if icons/images are fixed.

Pattern 6 (*Textual Annotation*)

Description This pattern denotes the availability of features to visually represent free-form text in the canvas, which can be attached to modeling elements without changing semantics.

Purpose To add domain-specific information (e.g. annotating an automated task with a text caption to explicate the task's inner working).

Realization BPMN is the only language providing a visual construct to attach free-form text to modeling elements called *Text Annotation*. This construct is supported by the main BPMN editors (see e.g. Intalio|Designer, ARIS and Oryx). Many modeling editors offers proprietary features to visualize free-form text, in order to compensate for those languages such as UML ADs and eEPCs, which do not support this pattern. For example, EA offers sticky notes for UML ADs, ARIS and Protos have text areas while Oryx has Text Notes for eEPCs.

Example The model in Fig 7 is also annotated with text captions to highlight those tasks that require access to an SAP system, to list all possible mismatches, and to indicate the procedure to follow in case of blocked invoices (all with explicative purpose).

Evaluation Criteria A language achieves full support if it has a construct that satisfies the pattern description. An implementation achieves full support if it supports the language's visual construct or provides proprietary features.

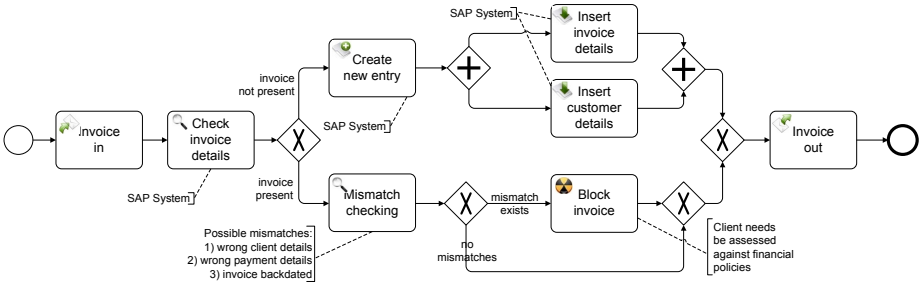

Fig. 7. Examples of Pictorial and Textual Annotations for the model in Fig. 8b.

Pattern 7 (*Explicit Representation*)

Description This pattern denotes the ability to capture process modeling concepts via a dedicated graphical notation.

Purpose To visualize and distinguish the various ingredients of a process model, not necessarily through a single view.

Rationale Explicit representation can reduce the cognitive overhead of associating syntactic elements to their semantics [22].

Realization The majority of process modeling languages provide graphical notations for a subset of their concepts only. In UML ADs, *AddStructuralFeatureValueAction* and *ApplyFunctionAction* are two examples of concepts that are only represented textually. Similarly, in BPMN 1.2 the various task types (e.g. Receive, Service, Manual), and the difference between *Embedded* and *Reusable* sub-process, are two examples of concepts that can only be distinguished via a task's textual attribute. Although these concepts have now been given a graphical notation in BPMN 2.0, still there are numerous element attributes that do not have one. In YAWL none of the concepts related to data and resourcing aspects are visually represented. In Protos joins and splits are always subsumed by an activity's multiple incoming, resp., outgoing edges. This is the same in Petri Nets for AND joins and splits. Only a few languages such as eEPCs and Workflow Nets, have a graphical notation for all their modeling concepts, although they feature less of them. A third class of languages including BPEL, XPDL and languages from the past such as BPML and XLANG, does not have a graphical notation. In the case of BPEL, the majority of editors provide a proprietary graphical notation (see e.g. JDeveloper or the Eclipse BPEL Editor), while others provide a BPMN skin to a BPEL model (e.g. Intalio|Designer). Out of the examined tools, Protos is the only one providing three different views of a process model in the canvas, such as the control-flow, the involved data and human roles.

Example The models in Fig 3-7 are all examples of process models whose modeling concepts (task, gateway, events, sequence flow) are explicitly represented via a dedicated graphical notation.

Evaluation Criteria A language achieves full support if all its concepts have a dedicated graphical notation. It rates as partial support if some concepts are implied and do not have dedicated graphical notation. An implementation achieves full support if it supports all the language graphical notations or provides proprietary notations to compensate for the lack of explicit graphical representation.

Pattern 8 (*Naming Guidance*)

Description This pattern refers to the availability of naming conventions or advice for model elements' labels, which can be syntactic (e.g. using a verb-object style) or semantic (e.g. using a domain-specific vocabulary).

Purpose To bring clarity and convey domain-specific information.

Rationale Names that follow a verb-object style are less ambiguous [26]. Names that better convey the modeler's intention improve understanding [8].

Realization None of the languages examined provides naming conventions or advice. Tool-wise, renaming features for task and process labels are explored (but not implemented) in [38], as part of a set of refactoring mechanisms for process models. A first effort towards the automation of renaming mechanisms is made in [7], where a prototype implementation is shown that can enforce specific naming conventions for eEPC elements, via thesauri and linguistic grammars. However, major tools still neglect naming guidelines, the only exception being made by the ARIS documentation [11] which indicates general semantic guidelines for eEPCs (e.g. avoiding generic verbs such as "to process"). On the other hand, the problem of establishing naming conventions for task names in process models has gained growing attention in academia and in the industry. From a syntactic perspective, Mendling et al. [26] conducted a systematic study of different syntactic styles for task names in process models. The result is that task names in the verb-object style are perceived as less ambiguous and more useful than names in other styles (e.g. action-noun). The use of the verb-object style for task names is also proposed as a modeling guideline in [27] and in [36]. Silver [36] also proposes naming guidelines for gateways and certain types of events in BPMN 2.0. From a semantic perspective, Becker et al. [8] envisage using a *business term catalogue* to establish and relate the main terms in an organization, which can be filtered depending on a specific user group. Rosemann [34] further develops this idea and recommends a preparatory step to process modeling where the involved terms are separately captured in a hierarchy with their semantic relations. Using a controlled vocabulary taken from a domain-specific reference model is suggested in [15], while in [26] the possibility of using a general data dictionary to control the object part of verb-object names is also envisaged. Regarding the verb part, Mendling et al. [24] propose a set of 25 frequently occurring verbs of general use, which they extracted from the SAP R/3 reference model [10] and generalized via established verb taxonomies.

Example Fig. 8 shows the model in Fig. 3b after renaming all activity labels in the verb-object style.

Selection Criteria A language achieves full support if it provides naming conventions or advice. An implementation achieves full support if it provides renaming capabilities that either follow the language naming conventions/advice or add proprietary extensions. It rates as partial support if it only provides naming conventions/advice.

4 Benchmarking

We report the results of evaluating a number of languages and tools against their support for the identified patterns. The criteria used are those presented in Section 3. In particular, for a tool the rationale was to measure the extent by which it facilitates the support for a pattern, as it is offered by a language. For

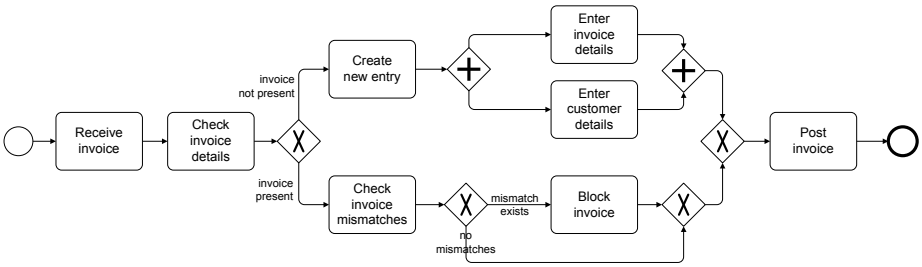

Fig. 8. Renaming the activity labels in Fig. 3 according to the verb-object style.

example, for Splitting, we checked if an editor allows one to jump between the splitting constructs provided by the language. The languages selected for this evaluation are mainstream process modeling languages deriving from standardization efforts, large-scale adoptions or established research initiatives. Specifically, we selected four languages for conceptual process modeling (UML ADs 2.1.1, eEPCs, BPMN 1.2 and BPMN 2.0) and four languages for executable process modeling (BPMN 2.0, BPEL 1.2/2.0, YAWL 2.0 and Protos 8.0.2). For each language, we also evaluated at least one supporting modeling editor. For UML ADs 2.1.1 we evaluated EA 7.1; for eEPCs and BPMN 1.2 we evaluated ARIS 7.1 and Oryx 2.0 beta; for BPEL 1.2 JDeveloper 11.1.1.1.0; for YAWL 2.0 the YAWL Editor 2.0 and for Protos the Protos editor 8.0.2. We did not evaluate any editor for BPMN 2.0 since at the time of writing there was no mature implementation for this specification. Table 9 shows the results of the analysis, where tool evaluations are shown next to the evaluations of their languages, except for Protos, where the language cannot be separated from its implementation.

	UML AD 2.1	Enterprise Architect 7.5	eEPCs	ARIS 7.1 (eEPCs)	Oryx 2.0 (eEPCs)	BPMN 1.2	ARIS 7.1 (BPMN 1.2)	Oryx 2.0 (BPMN 1.2)	BPMN 2.0	BPEL 1.2/2.0	JDeveloper 11.1.1.1.0	YAWL 2.0	YAWL Editor 2.0	Protos 8.0.2
1 Layout Guidance	-	-	+	+	-	+	+	-	+	-	+	-	+/-	-
2 Layout Split	+	+/-	+	+	+/-	+	+	+ /-	+	-	-	-	-	-
3 Group Highlight	-	+	-	+	-	+	+	+	+	-	-	-	-	+
4 Graphical Highlight	-	+	+	+	+	+	+	+	+	-	+/-	-	+	-
5 Pictorial Annotation	-	+	-	-	-	-	-	-	+	-	+/-	-	+	+/-
6 Textual Annotation	-	+	-	+	+	+	+	+	+	-	-	-	-	+
7 Explicit Representation	+/-	+	+	+	+	+/-	+	+	+/-	-	+	+/-	+	+/-
8 Naming Guidance	-	-	-	+/-	-	-	-	-	-	-	-	-	-	-

Fig. 9. Evaluation results.

From the table, we can make the following observations. First, the selected tools generally offer wider pattern support than the respective languages. Examples are the differences between EA and UML ADs, between eEPCs and ARIS for eEPCs, and between YAWL and the YAWL Editor. A possible reason is that languages typically focus on defining the process syntax and semantics, but not

on visualization features that are convenient in a modeling environment. When implementing support for these languages in a modeling editor, visualization features become a major concern. Clearly, language support being equal, the more sophisticated visualization features an editor can offer, the more competitive it is on the market. Second, the tools that are primarily developed for conceptual process modeling provide better patterns support than those developed for executable process modeling. For example, ARIS for BPMN fully supports six patterns and partially supports one, while Protos offers full support for only two patterns, and partial support for other two patterns. This can be explained by the fact that the visualization features in the second class of tools are not the main focus, as opposed to other features such as data specification and application integration. Third, we can observe an increase in patterns support from UML ADs to eEPCs, BPMN 1.2 and finally to BPMN 2.0, which clearly reflects the evolution of process modeling languages. On the other hand, BPEL is the only language that does not support any pattern. This is justified by the fact that BPEL does not define an official graphical notation. Forth, the limited support for Pictorial Annotation can be explained by the recent advances in computer graphics—pictorial annotations were not possible a decade ago—and the growing need for decorating process models with attributes familiar to business users. Finally, the even less support for Naming Guidance derives from the fact that traditionally the development of modeling languages has not been concerned with the use of linguistic support such as ontologies. However, we can observe a growing academic interest in this pattern from the literature.

5 Conclusion

In this paper, we conducted a systematic analysis of features that exist for managing complexity of process models, where these features affect the concrete syntax but not the abstract syntax of a model. The result of the analysis took the form of a collection of patterns and an evaluation of state-of-the-art languages and language implementations in terms of these patterns. While one cannot prove that the patterns collection is complete (as there is no reference framework that could be used for this purpose), confidence about the comprehensiveness of this patterns collection is derived from the careful consideration of the relevant literature, standards and tools.

This pattern-based analysis of the state-of-the-art in process modeling, identified relative strengths and weaknesses among the languages and tools considered. This analysis may provide a basis for further language and tool development. For example, contemporary tools could provide support for naming conventions or guidelines from both a syntactic and a semantic perspective.

It is worthwhile to conduct further research into the way the various features presented in this paper can be best used. In particular, it should be considered how multiple features can be combined to provide a better understanding of a process model. Ideally this would involve the conduct of extensive empirical

studies. Subsequent work is planned to look at features and approaches for managing process model complexity which affect the abstract syntax of a process model.

References

1. W.M.P. van der Aalst. TomTom for Business Process Management (TomTom4BPM). In *CAiSE*, volume 5565 of *LNCS*, pages 2–5. Springer, 2009.
2. 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.
3. W.M.P. van der Aalst and K.B. Lassen. Translating unstructured workflow processes to readable BPEL: Theory and implementation. *Inf. Softw. Technol.*, 50(3):131–159, 2008.
4. W.M.P. van der Aalst and K. M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT Press, 2002.
5. A.A. Abdul, G.K.T. Wei, G.M. Muketha, and W.P. Wen. Complexity Metrics for Measuring the Understandability and Maintainability of Business Process Models using Goal-Question-Metric (GQM). *Int. Journal of Computer Science and Network Security*, 8(5):219–225, 2008.
6. R. Alpfelbacher, A. Knopfel, P. Aschenbrenner, and S. Preetz. FMC Visualization Guidelines. `http://www.fmc-modeling.org/visualization_guidelines`, 2006. Accessed: Nov 2009.
7. J. Becker, P. Delfmann, S. Herwig, L. Lis, and A. Stein. Towards increased comparability of conceptual models – enforcing naming conventions through domain thesauri and linguistic grammars. In *Proceedings of ECIS*, 2009.
8. J. Becker, M. Rosemann, and C. von Uthmann. Guidelines of Business Process Modeling. In *Business Process Management*, volume 1806 of *LNCS*, pages 30–49. Springer, 2000.
9. J. Cardoso, J. Mendling, G. Neumann, and H.A. Reijers. A Discourse on Complexity of Process Models. In *Business Process Management Workshops*, volume 4103 of *LNCS*, pages 117–128. Springer, 2006.
10. T. Curran and G. Keller. *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*. Upper Saddle River, 1997.
11. R.B. Davis. *Business Process Modelling with ARIS: A Practical Guide*. Springer, 2001.
12. A. Eckleder, T. Freytag, J. Mendling, and H.A. Reijer. Realtime Detection and Coloring of Matching Operator Nodes in Workflow Nets. In *Algorithms and Tools for Petri Nets*, pages 56–61. CEUR, 2009.
13. P. Effinger, M. Kaufmann, and M. Siebenhaller. Enhancing Visualizations of Business Processes. In *Graph Drawing*, volume 5417 of *LNCS*, pages 437–438. Springer, 2008.
14. P. Effinger, M. Siebenhaller, and M. Kaufmann. An Interactive Layout Tool for BPMN. *E-Commerce Technology*, 0:399–406, 2009.
15. R. Eshuis and P.W.P.J. Grefen. Constructing customized process views. *Data & Knowledge Engineering*, 64(2):419–438, 2008.
16. C. W. Günther and W.M.P. van der Aalst. Fuzzy Mining - Adaptive Process Simplification Based on Multi-perspective Metrics. In *Business Process Management*, volume 4714 of *LNCS*, pages 328–343. Springer, 2007.
17. J. Huotari, K. Lyytinen, and M. Niemelä. Improving graphical information system model use with elision and connecting lines. *ACM TCHI*, 11(1):26–58, 2004.

18. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use.* EATCS monographs on Theoretical Computer Science. 1996.
19. G. Keller, M. Nüttgens, and A.-W. Scheer. Semantische Processmodellierung auf der Grundlage Ereignisgesteuerter Processketten (EPK). Technical report, University of Saarland, Germany, 1992 (in German).
20. I. Kitzmann, C. König, D. Lübke, and L. Singer. A Simple Algorithm for Automatic Layout of BPMN Processes. volume 0, pages 391–398. IEEE Computer Society, 2009.
21. R. Laue and V. Gruhn. *Technologies for Business Information Systems*, chapter Approaches for Business Process Model Complexity Metrics, pages 13–24. Springer, 2007.
22. G.L. Lohse. A Cognitive Model for Understanding Graphical Perception. *Human-Computer Interaction*, 8:353–388, 1993.
23. J. Mendling, B.F. van Dongen, and W.M.P. van der Aalst. Getting rid of OR-joins and multiple start events in business process models. *Ent. IS*, 2(4):403–419, 2008.
24. J. Mendling, J. Recker, and H.A. Reijers. On The Usage of Labels and Icons in Business Process Modeling. *Int. Journal of Information System Modeling and Design*, 2009. to appear.
25. J. Mendling, H.A. Reijers, and J. Cardoso. What Makes Process Models Understandable? In *Business Process Management*, volume 4714 of *LNCS*, pages 48–63. Springer, 2007.
26. J. Mendling, H.A. Reijers, and J. Recker. Activity Labeling in Process Modeling: Empirical Insights and Recommendations. *Information Systems*, 2009. to appear.
27. J. Mendling, H.A. Reijers, and W.M.P. van der Aalst. Seven Process Modeling Guidelines (7PMG). *Information and Software Technology*, 52(2):127–136, 2010.
28. J. Mendling and M. Strembeck. Influence Factors of Understanding Business Process Models. In *Business Information Systems*, LNBIP, pages 142–153. Springer, 2008.
29. OMG. *Business Process Model and Notation (BPMN), ver. 2.0 (draft)*, May 2009. http://www.bpmnstyle.com/wp-content/uploads/BPMN%202-0%20Specification%20BMI2009-05-03.pdf.
30. OMG. *Business Process Modeling Notation (BPMN), ver. 1.2*, January 2009. http://www.omg.org/docs/formal/09-01-03.pdf.
31. A. Paivio. Dual Coding Theory: Retrospect and Current Status. *Canadian Journal of Psychology*, 45(3):255–287, 1991.
32. M.D.L. Proano. Visual Layout for Drawing Understandable Process Models. Master's thesis, Eindhoven University of Technology, 2008.
33. H.C. Purchase. Which Aesthetic has the Greatest Effect on Human Understanding? In *Graph Drawing*, volume 1353 of *LNCS*, pages 248–261. Springer, 1997.
34. M. Rosemann. *Process Management: A guide for the design of business processes*, chapter Preparation of process modeling, pages 41–78. Springer, 2003.
35. M. Schrepfer, J. Wolf, J. Mendling, and H.A. Reijers. The Impact of Secondary Notation on Process Model Understanding. In *PoEM*, pages 161–175. IFIP, 2009.
36. B. Silver. *BPMN Method & Style.* Cody-Cassidy Press, 2009.
37. A. Streit, B. Pham, and R. Brown. Visualization Support for Managing Large Business Process Specifications. In *BPm*, pages 205–219, 2005.
38. B. Weber and M. Reichert. Refactoring Process Models in Large Process Repositories. In *CAiSE*, volume 5074 of *LNCS*. Springer, 2008.
39. M. Weske. *Business Process Management: Concepts, Languages, Architectures.* Springer, 2007.