

QUT Digital Repository:
<http://eprints.qut.edu.au/>



La Rosa, Marcello and ter Hofstede, Arthur H.M. and Wohed, Petia (2010)
Managing process model complexity via concrete syntax modifications.

Copyright 2010 the Authors

Managing Process Model Complexity via Concrete Syntax Modifications

Marcello La Rosa¹, Arthur H.M. ter Hofstede^{1,2}, and Petia Wohed³

¹ Queensland University of Technology, Australia
{m.larosa, a.terhofstede}@qut.edu.au

² Eindhoven University of Technology, Netherlands

³ Stockholm University, 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 (i.e. with its concrete syntax), and 2) those that in essence change the structure of the model (i.e. its abstract syntax). 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 [50]. 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 dependencies among them. The more complex a business process model is, the harder it is to use it in the communication with 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 changing business practices).

There is a substantial body of literature on process model complexity and understandability on the one hand (e.g. [9, 26, 30, 33, 4, 44]) as well as on proposed mechanisms to deal with managing this complexity on the other hand (e.g. [46, 49, 28, 2]). However, what is lacking is a language-independent 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. Specifically, it may benefit different process model stakeholders, including those designing and standardizing process modeling languages such as BPMN, those developing modeling tools to support such languages, those currently using a specific language/tool in order to evaluate its strengths and weaknesses, and those who plan to do so.

In this paper we follow the established approach of capturing a comprehensive range of desired capabilities through a collection of patterns (e.g. the workflow patterns [51] provide a language-independent description of expressiveness requirements in process modeling languages). The patterns capture features to manage process model complexity as they are found in the literature, in process modeling languages or in their tool implementations.

In line with the field of programming languages [34], we distinguish between *concrete syntax* and *abstract syntax* of a model. The symbols used to represent the various types of nodes in a process model such as tasks, events, connectors, roles, 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 captures the various types of process elements and the structural 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. Accordingly, we classify features to reduce the complexity of a process model into two categories: those that affect the concrete syntax of a model only, and those that also affect the abstract syntax. In this paper, we focus on complexity reduction features that affect a model's concrete syntax only.

The patterns description is language-independent, but typically the realization of these patterns in one or more existing approaches is used to reinforce understanding and to demonstrate relevance. This description is complemented by an overview of the degree of patterns support in a number of well-known process modeling languages and language implementations, which provides an insight into comparative strengths and weaknesses.

The rest of this paper is structured as follows. Section 2 presents the various patterns while Section 3 evaluates the support offered by mainstream BPM languages and tools for the identified patterns. Next, Section 4 discusses related work and Section 5 concludes the paper.

2 Patterns for Concrete Syntax Modification

In order to establish a comprehensive collection of patterns, we extensively analyzed state-of-the-art techniques and approaches reported in the BPM literature, we studied the specifications of various process modeling languages, experimented with a wide set of commercial and open source BPM tools, and incorporated feedback from researchers in the field. As a result of this study, we identified nine patterns operating exclusively on the concrete syntax of a process model and classified them according to the hierarchy in Fig. 1. 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 *Enclosure Highlight*, *Graphical Highlight*, and two annotation patterns: *Pictorial Annotation* and *Textual Annotation*. Two representation patterns, *Explicit Representation* and *Shorthand Representation*, refer to the availability of explicit and shorthand visual representations for process modeling constructs. Finally, *Naming Guidance* refers to naming conventions or advice to be used in a process model.

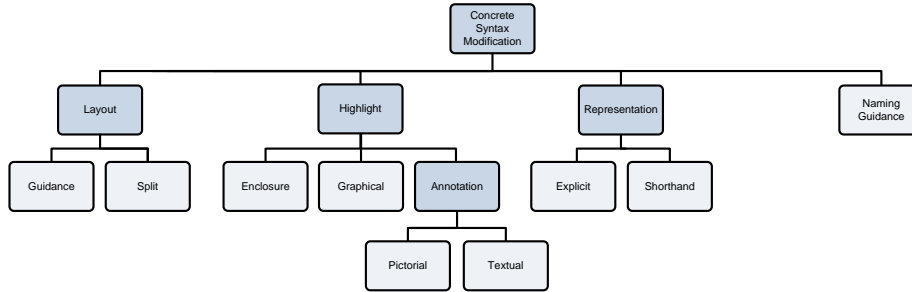


Fig. 1. Patterns for concrete syntax modification.

In order to illustrate the above patterns, we use the BPMN (Business Process Modeling Notation) standard [37]. An overview of the graphical representation of the main concepts of this notation can be found in Fig. 2. 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 [37].

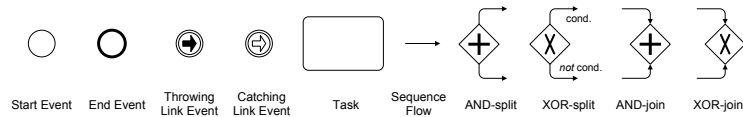


Fig. 2. BPMN 1.2 modeling elements used in this paper.

In the following we provide a detailed description of each pattern including its purpose, the rationale for its use, its occurrence in languages, tools and in the literature, and an example of its realization in BPMN.

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 orientation, alignment and spacing of model 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 [30]. Crossing edges negatively affect model understanding [40].

Realization Some languages provide general guidelines on how a model should be laid out on the canvas. eEPCs [21, 11] prescribe to model processes from top to bottom [21], 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” [37], p.30. Tool-wise, we can distinguish three categories. Some tools offer algorithms to lay out process models. These can be very sophisticated, such as in the case of Software AG ARIS which supports both eEPCs and BPMN layout guidelines, and where elements orientation, alignment and spacing can be customized, or simple ones, such as in the case of the YAWL Editor. Other tools, such as Oracle JDeveloper, impose a fixed layout. A third category which includes the Sparx Enterprise Architect (EA) and Pallas Athena Protos, provides limited to 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. [5] suggest to place related elements spatially close to each other, Huotari et al. [19] and Purchase [40] point out that crossing edges should be avoided if possible, while Jensen [20] 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 [22], 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 [44].

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.

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 (without implications to the semantics) or purely necessitated by physical constraints, such as the length of an edge, the size of the modeling canvas or of the printing page.

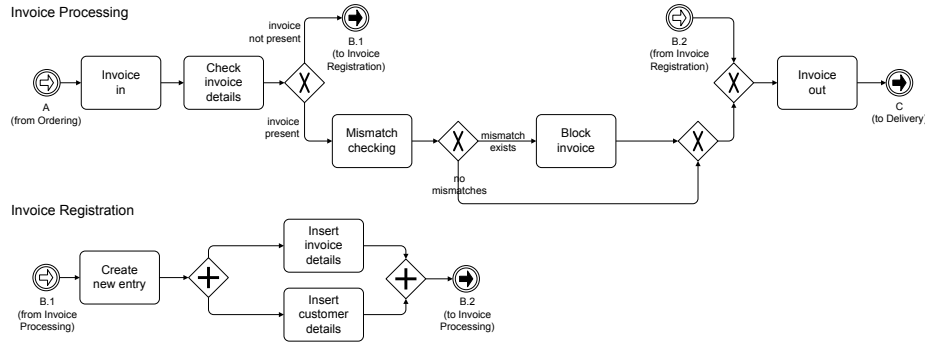


Fig. 4. Two examples of the use of Layout Split on the model in Fig. 3b.

Pattern 3 (Enclosure Highlight)

Description This pattern refers to the availability of modeling constructs to visually enclose 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. enclosing all elements that need revision or all elements that refer to a given resource.

Rationale Visually enclosing a set of elements increases the perceptual discriminability of the enclosed elements from the others [23].

Realization As per languages, BPMN is the only one that supports this pattern via the notion of Grouping—a dashed-line, rounded corner rectangle with a name, used to enclose a set of model elements. The elements in a BPMN Grouping are only grouped informally. The majority of modeling editors provide a drawing palette to allow drawing a shape to enclose 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 provide comments. EA offers a non-UML element called System Boundary to define conceptual boundaries from a visual perspective. Visual enclosure has been recognized as a means to discriminate among visual elements already by the Gestalt psychologists in 1935 [23], via the perceptual relationship of Containment. However, to our knowledge the use of this mechanism has not yet been investigated in the context of business process modeling.

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.

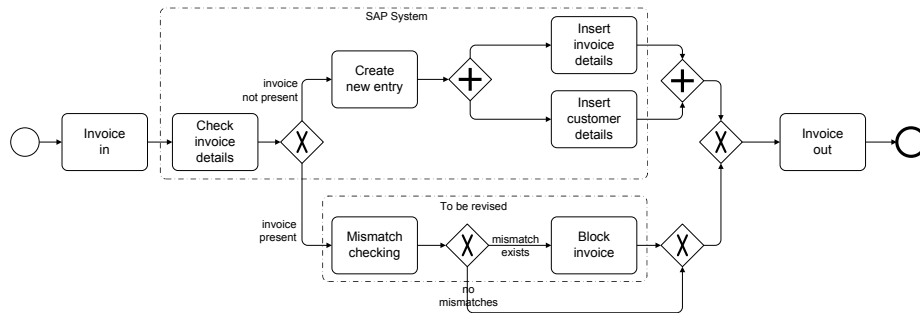


Fig. 5. An example of Enclosure Highlight using the BPMN Grouping construct.

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 Using a range of appearance properties results in a perceptually enriched representation that can reduce the cognitive overhead of associating syntactic elements with their semantics [27, 35].

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. 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 [39], 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 [17] 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.

Pattern 5 (Pictorial Annotation)

Description This pattern denotes the availability of features to assign pictorial elements, such as icons or images, to modeling elements.

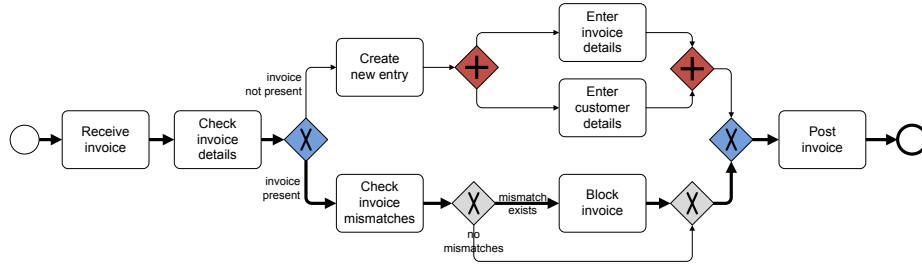


Fig. 6. Two examples of Graphical Highlight: coloring and line thickening.

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 [38] by speeding up recognition and recall, especially for naive users [35].

Realization In BPMN 2.0 [36], 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. [29] 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).

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).

Rationale Textual annotations can improve understanding of diagrams as comments can improve understanding of source code [35].

Realization UML and BPMN provide a visual construct to attach free-form text to modeling elements called, respectively, *Comment* and *Text Annotation*. This construct is supported by the main UML and BPMN editors (see e.g. EA, 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 eEPCs which do not support this pattern. For example, 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).

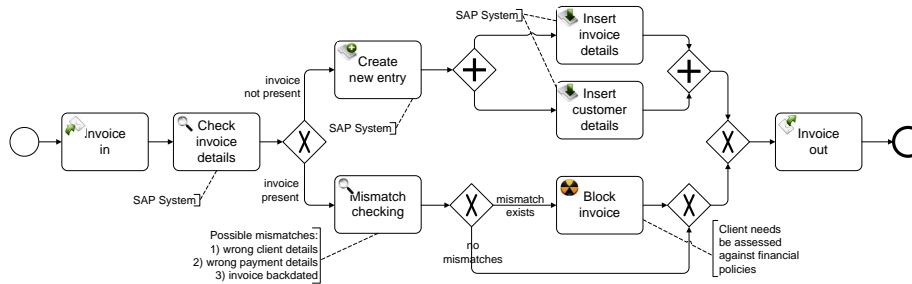


Fig. 7. Examples of Pictorial and Textual Annotations for the model in Fig. 9b.

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.

Rationale Explicit representation can reduce the cognitive overhead of associating syntactic elements to their semantics [27].

Realization The majority of process modeling languages provide graphical notations for a subset of their concepts only. In UML ADs, *AddStructuralFeature ValueAction* 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 [18] 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 [3], have a graphical notation for all their modeling concepts,

although they feature less of them. A third class of languages including BPEL, XPD and languages from the past such as BPML and XLANG, does not have a graphical notation. In the case of BPEL, the majority of BPEL 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).

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.

Pattern 8 (Shorthand Representation)

Description This pattern denotes the ability to capture process modeling concepts without the use of their primary graphical notation.

Purpose To avoid cluttering and reduce model size, especially in large or complex models.

Rationale Reducing model size positively affects model understanding [30].

Realization BPMN offers a number of shorthand representations. For example, an AND-split can be replaced by specifying multiple outgoing edges from an activity; an (X)OR split can be replaced by *Conditional Flows*; an XOR-join can be subsumed by multiple incoming edges to an activity; an incoming/outgoing message flow can be directly connected to an activity (thus avoiding the use of a message event or a receive/send activity). UML ADs offer similar shorthand notations for the AND-split and the (X)OR split, while the AND-join is subsumed by multiple incoming edges to an activity. In YAWL conditions are always alternated with tasks. However they can be omitted from the graphical representation of a model when connected only to one preceding and to one subsequent task. These shorthand representations are generally supported by the main modeling editors, see e.g. EA for UML ADs, ARIS and Oryx for BPMN, and the YAWL Editor for YAWL.

Example Fig 8 shows the model in Fig 3b after applying the BPMN shorthand representation for XOR-split, XOR-join and AND-split.

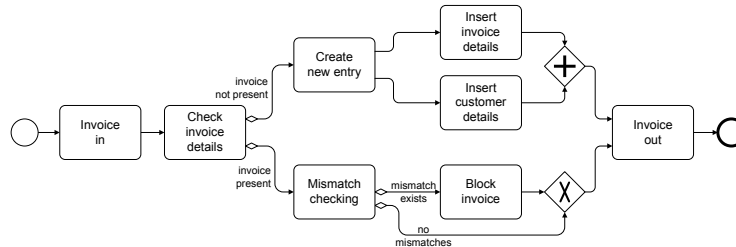


Fig. 8. Shorthand Representation of splits and joins for the model in Fig. 3b.

Pattern 9 (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 [31]. Names that better convey the modeler’s intention improve understanding [7].

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 [49], as part of a set of refactoring mechanisms for process models. A first effort towards the automation of renaming mechanisms is made in [6], 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. [31] 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 [32] and in [45]. Silver [45] also proposes naming guidelines for gateways and certain types of events in BPMN 2.0. From a semantic perspective, Becker et al. [7] 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 [42] 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 [31] 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. [29] 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. 9 shows the model in Fig. 3b after renaming all activity labels in the verb-object style.

3 Benchmarking

We report the results of evaluating a number of languages and tools against their support for the identified patterns. The languages selected for this evaluation are mainstream process modeling languages deriving from standardization

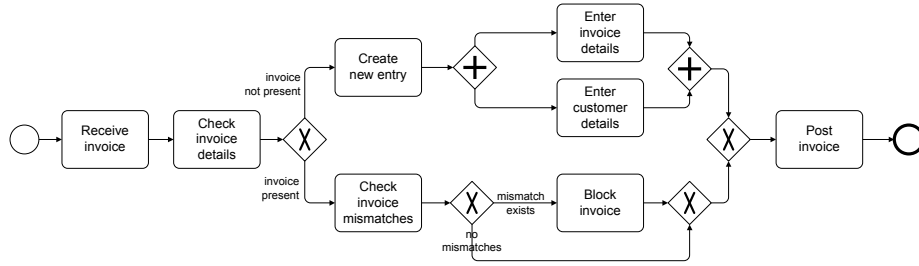


Fig. 9. Renaming the activity labels in Fig. 3 according to the verb-object style.

efforts, large-scale adoptions or established research initiatives. 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 8.0.2¹). 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 BPMN 2.0 we evaluated 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. Table 10 shows the results of this 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. 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. Accordingly, for Layout Guidance, we decided to rate as ‘-’ all tools providing limited layout support (e.g. alignment only). For Splitting we rated EA and Oryx ‘+/-’ as they do not allow jumping between the splitting constructs provided by the language. For Graphical Highlight, we rated JDeveloper ‘+/-’ as the appearance of model elements cannot be customized. For Pictorial Annotation, JDeveloper and Protos received a ‘+/-’ as default icons or images are automatically assigned to model elements and cannot be customized. For Naming Guidance, ARIS received a ‘+/-’ as it provides general semantic guidelines for eEPC labels but does not offer renaming capabilities to enforce these guidelines. Regarding languages evaluation, we rated UML ADs 2.1, BPMN 1.2/2.0, YAWL 2.0 and Protos 8.0.2 ‘+/-’ for Explicit Representation, as they do not have a graphical representation for some concepts.

From the table we can make the following observations. First, as expected, the selected tools generally offer wider pattern support than the respective languages. Examples are the differences between UML ADs and EA, 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,

¹ Technically Protos specifications can be executed, although in practice this language is mainly used for process modeling

| | 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 1.2/2.0) | 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 Enclosure Highlight | - | + | - | + | - | + | + | + | + | - | - | - | - | + |
| 4 Graphical Highlight | - | + | + | + | + | + | + | + | + | - | +/- | - | + | - |
| 5 Pictorial Annotation | - | + | - | - | - | - | - | - | + | - | +/- | - | + | +/- |
| 6 Textual Annotation | - | + | - | + | + | + | + | + | + | - | - | - | - | + |
| 7 Explicit Representation | +/- | + | + | + | + | +/- | + | + | +/- | - | + | +/- | + | +/- |
| 8 Shorthand Representation | + | + | - | - | - | + | + | + | + | - | - | + | + | - |
| 9 Naming Guidance | - | - | - | +/- | - | - | - | - | - | - | - | - | - | - |

Fig. 10. Evaluation results.

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 seven patterns, while Protos offers full support for two patterns only, 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, role allocation and application integration. Third, we observe an increase in patterns support from UML ADs to eEPCs, BPMN 1.2 and finally to BPMN 2.0. This 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. Nonetheless, we included BPEL in our analysis for completeness. Fourth, 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, as evidenced by this pattern’s realization.

4 Related Work

Three main benchmarking frameworks have been used to evaluate process modeling languages: the workflow patterns framework [51], Bunge, Wand and Weber’s

(BWW) framework [48], and the Semiotic Quality Framework (SEQUAL) [24]. The workflow patterns provide a language-independent description of control-flow, resource, data and exception handling aspects in workflow languages. Their development started as a bottom-up, comparative analysis of process modeling languages and tools, with the purpose to evaluate their suitability and determine similarities and differences between them. To date, the workflow patterns have been used to examine the capabilities of numerous process modeling/workflow languages, standards and tools [51]. Our work is complementary to this framework since it describes recurring features (patterns) to reduce the complexity of process models, and can be used for the evaluation of process modeling languages and tools.

The BWW framework refers to Wand and Weber’s tailoring and application of Bunge’s ontology [8] to information systems. It was initially used for analysis and comparison of conceptual modeling languages, and later also used for the analysis of process modeling languages [43, 41]. However, the BWW framework lacks conceptual structures central to process modeling such as various types of splits and joins, iteration and cancelation constructs, and different forms of concurrency restrictions. Thus, despite its utilization in practice, its suitability for evaluating process modeling languages can be questioned. Also, its application as a theoretical foundation for conceptual modeling has been criticized [52].

The SEQUAL framework introduces and reasons about different aspects relevant to model quality. These aspects span different quality notions, including physical, empirical, syntactic, semantic, pragmatic and social quality. Particularly relevant to our work are the empirical quality, which deals with readability matters such as graph aesthetics, and the pragmatic quality, which deals with the understanding of a model by its audience. SEQUAL has been used for the evaluation of process modeling languages [47], and has later been extended to deal specifically with quality of process models [25]. Nonetheless, the authors themselves acknowledge SEQUAL’s “disability (sic) to facilitate precise, quantitative evaluations of models” [25], p.101. In contrast, our patterns collection provides a concrete means to evaluate the pragmatic and empirical quality of process modeling languages, and can also be applied to evaluate supporting tools.

Our work can also be related to [35], where a theory of general principles for designing cognitive-effective visual notations is proposed. Specifically, our patterns can be seen as an implementation of the Complexity Management principle, which recommends that a visual notation should include explicit mechanisms to simplify a model’s appearance. Moreover, the Textual Annotation pattern can be seen as an implementation of the Dual Coding principle, which prescribes the use of text to complement graphics, while the Graphical Highlight pattern can be seen as an implementation of the Semantic Transparency principle, which prescribes the use of visual representations whose appearance suggests their meaning.

5 Conclusion

The main contribution of this paper is a systematic analysis of concrete syntax modifications for reducing process model complexity, as they occur in the literature, in process modeling languages and tool implementations. The result of this 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. This collection can be useful for different process model stakeholders, including those designing and standardizing process modeling languages (such as BPMN and UML ADs), those developing modeling tools to support such languages, those currently using a specific language/tool in order to evaluate its strengths and weaknesses, and those who plan to do so.

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 a careful survey of the relevant literature, standards and tools for process modeling. Although some of these patterns may be applied to other types of models, e.g. data models, our focus was solely on process models.

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 support naming conventions or guidelines, from both a syntactic and a semantic perspective, or more, allow modelers to easily switch between shorthand notations and their full expansions depending on user preferences.

The evaluation reported in this paper shows whether or not a given pattern is supported by the various tools. However, one may be interested in a more accurate evaluation of the degree of support for each pattern, especially for tool selection purposes. For example, the majority of tools offer layout algorithms. Still, some of these algorithms (e.g. the one in ARIS) provide better results in terms of elements alignment, distribution and spacing than others. In order to determine finer grained pattern support, we plan to conduct experiments with end users to compare the perceived model understandability after using different tool features (e.g. different layout algorithms). Another aspect worth investigating through experimentation is under which process model characteristics multiple patterns can be combined to increase process model understanding. In fact, there might be cases in which the application of some combinations of patterns may actually decrease the understanding of a process model, e.g. using pictorial annotation with graphical highlight.

The patterns presented in this paper describe features that affect a model's concrete syntax only. We are currently studying those features that also affect a model's abstract syntax, e.g. structuring a model, modularizing a model into subprocesses or peer processes, collapsing or repeating model elements.

Acknowledgements. We would like to thank Hajo Reijers for his valuable feedback on an earlier version of this paper.

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 and K.B. Lassen. Translating unstructured workflow processes to readable BPEL: Theory and implementation. *Inf. Softw. Technol.*, 50(3):131–159, 2008.
3. W.M.P. van der Aalst and K. M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT Press, 2002.
4. 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.
5. R. Alpfelbacher, A. Knopfel, P. Aschenbrenner, and S. Preetz. FMC Visualization Guidelines. http://www.fmc-modeling.org/visualization_guidelines, 2006. Accessed: Nov 2009.
6. 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.
7. 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.
8. M. Bunge. *Treatise on Basic Philosophy Volume 3: Ontology I – The Furniture of the World*. Kluwer Academic Publishers, 1977.
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. Reijers. 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. Object Management Group. *OMG Unified Modeling Language Superstructure 2.2*. OMG, <http://www.omg.org/spec/UML/2.2>, 2009.
17. 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.
18. A.H.M. ter Hofstede, W.M.P. van der Aalst, M. Adams, and N. Russell. *Modern Business Process Automation: YAWL and its Support Environment*. 2010.
19. 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.
20. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*. EATCS monographs on Theoretical Computer Science. 1996.

21. G. Keller, M. Nüttgens, and A.-W. Scheer. Semantische Prozessmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK). Technical report, University of Saarland, Germany, 1992 (in German).
22. I. Kitzmann, C. König, D. Lübke, and L. Singer. A Simple Algorithm for Automatic Layout of BPMN Processes. In *Proc. of IEEE Conference on Commerce and Enterprise Computing*, pages 391–398. IEEE Computer Society, 2009.
23. K. Koffka. *Principles of Gestalt Psychology*. Harcourt, Brace and Co., 1935.
24. J. Krogstie, O.I. Lindland, and G. Sindre. Defining quality aspects for conceptual models. In *Proc. of ISCO*, pages 216–231, 1995.
25. J. Krogstie, G. Sindre, and H. Jorgensen. Process models representing knowledge for action: a revised quality framework. *Eur. J. Inf. Syst.*, 15(1):91–102, 2006.
26. R. Laue and V. Gruhn. *Technologies for Business Information Systems*, chapter Approaches for Business Process Model Complexity Metrics, pages 13–24. Springer, 2007.
27. G.L. Lohse. A Cognitive Model for Understanding Graphical Perception. *Human-Computer Interaction*, 8:353–388, 1993.
28. 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.
29. 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*, 1(2):40–58, 2009.
30. 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.
31. J. Mendling, H.A. Reijers, and J. Recker. Activity Labeling in Process Modeling: Empirical Insights and Recommendations. *Information Systems*, 35(4):467–482, 2010.
32. 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.
33. J. Mendling and M. Strembeck. Influence Factors of Understanding Business Process Models. In *Business Information Systems*, LNBIP, pages 142–153. Springer, 2008.
34. B. Meyer. *Introduction to the Theory of Programming Languages*. Prentice-Hall, Englewood Cliffs, New Jersey, 1990.
35. D.L. Moody. The “Physics” of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. *IEEE Transactions on Software Engineering*, 35:756–779, 2009.
36. 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>.
37. OMG. *Business Process Modeling Notation (BPMN), ver. 1.2*, January 2009. <http://www.omg.org/docs/formal/09-01-03.pdf>.
38. A. Paivio. Dual Coding Theory: Retrospect and Current Status. *Canadian Journal of Psychology*, 45(3):255–287, 1991.
39. M.D.L. Proano. Visual Layout for Drawing Understandable Process Models. Master’s thesis, Eindhoven University of Technology, 2008.
40. H.C. Purchase. Which Aesthetic has the Greatest Effect on Human Understanding? In *Graph Drawing*, volume 1353 of *LNCS*, pages 248–261. Springer, 1997.
41. J.C. Recker, M. Indulska, M. Rosemann, and P. Green. How Good is BPMN Really? Insights from Theory and Practice. In *Proc. of ECIS*, 2006.
42. M. Rosemann. *Process Management: A guide for the design of business processes*, chapter Preparation of process modeling, pages 41–78. Springer, 2003.

43. M. Rosemann, J. Recker, M. Indulska, and P. Green. A Study of the Evolution of the Representational Capabilities of Process Modeling Grammars. In *Proc. of CAiSE*, pages 447–461, 2006.
44. 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.
45. B. Silver. *BPMN Method & Style*. Cody-Cassidy Press, 2009.
46. A. Streit, B. Pham, and R. Brown. Visualization Support for Managing Large Business Process Specifications. In *BPM*, volume 3649 of *LNCS*, pages 205–219. Springer, 2005.
47. T. Wahl and G. Sindre. *Advanced Topics in Database Research*, volume 5, chapter An Analytical Evaluation of BPMN Using a Semiotic Quality Framework. IGI Global, 2006.
48. Y. Wand and R. Weber. On the Ontological Expressiveness of Information Systems Analysis and Design Grammars. *Journal of Information Systems*, 3:217–237, 1993.
49. B. Weber and M. Reichert. Refactoring Process Models in Large Process Repositories. In *CAiSE*, volume 5074 of *LNCS*. Springer, 2008.
50. M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, 2007.
51. Workflow Patterns Initiative. Home Page. <http://www.workflowpatterns.com>. Accessed: June 2010.
52. B. Wyssusek. On Ontological Foundations of Conceptual Modelling. *Scandinavian Journal of Information Systems*, 18(1):63–80, 2006.