Managing Process Model Complexity via Concrete Syntax Modifications

Marcello La Rosa, Arthur H.M. ter Hofstede, Petia Wohed, Hajo A. Reijers, Jan Mendling and Wil M.P. van der Aalst, *Member, IEEE*

***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 the ever-growing complexity of business process models. Mechanisms for dealing with this complexity can be classified into two categories: i) those that are solely concerned with the visual representation of the model, and ii) those that change its inner structure. While significant attention is paid to the latter category in the BPM literature, this paper focuses on the former category. It presents a collection of patterns that generalize and conceptualize various existing mechanisms to change the visual representation of a process model. Next, it provides a detailed analysis of the degree of support for these patterns in a number of state-of-the-art languages and tools. The paper concludes with the results of a usability evaluation of the patterns conducted with BPM practitioners.**

***Index Terms*—Process model, pattern, complexity, presentation, secondary notation.**

I. Introduction

Business Process Management (BPM) deals with the life-cycle of business process models which includes their design, execution and analysis [60]. Through the application of BPM technology, businesses may realize cost reductions, time savings, and an increased agility to deal with change. Many organizations have been investing in this technology and the interest in BPM surged in recent years. Despite advancements in the field of BPM—both in academia and in industry—important challenges still remain. These need to be dealt with in order to fully realize the potential of BPM technology.

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 determine if it properly captures the right business practices, to use it to communicate with stakeholders, and to evolve it over time (e.g. due to unforeseen circumstances or changing business requirements).

There is a substantial body of literature on process model complexity and understandability on the one hand (e.g. [12], [32], [37], [40], [4], [53], [49]) as well as on proposed mechanisms to deal with managing this complexity on the other hand (e.g. [56], [59], [21]). 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. A variety of process model stakeholders may benefit from this, e.g., those designing and standardizing process modeling languages such as BPMN, those developing modeling tools to support such languages, and those currently using a specific language/tool in order to evaluate its strengths and weaknesses. These observations triggered the development of a language-independent overview of the various features that exist to reduce the complexity of a process model.

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 [61] 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 [41], we distinguish between *concrete syntax* and *abstract syntax* of a model. The concrete syntax of a process modeling language deals only with representational aspects such as symbols, colors and position, of the various types of nodes in a process model (e.g. tasks, events, gateways, roles). In cognitive sciences [46], this corresponds to the cognitive dimension of *secondary notation*. 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 symbol of a certain node type) should not have any consequences for its abstract syntax representation. Modularizing a process model in a hierarchy of nested subprocesses can be used to simplify a model without changing its behavior. However, modularization affects the abstract syntax. Abstracting from certain elements of the model (e.g. in order to create a process view for a specific class of users) also affects

Manuscript received February 6, 2011; revised March 2, 2011.

M. La Rosa and A.H.M. ter Hofstede are with the Queensland University of Technology. e-mail: see http://marcellolarosa.com and http://yawlfoundation.org/∼arthur. ter Hofstede is also with Eindhoven University of Technology.

P. Wohed is with Stockholm University, Sweden. e-mail: see http://people.dsv.su.se/∼petia.

H.A. Reijers and W.M.P. van der Aalst are with Eindhoven University of Technology, The Netherlands. e-mail: see http://www.reijers.com and http://www.vdaalst.com. van der Aalst is also with Queensland University of Technology.

J. Mendling is with Humboldt University of Berlin, Germany. e-mail: see http://mendling.com.

the abstract syntax of a process model—in fact certain model elements are replaced or removed altogether. 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 primarily affect its abstract syntax*, and, as a consequence, may also impact on its concrete syntax. In this paper, we focus on complexity reduction features that only affect the concrete syntax of a process model.

The patterns description is language-independent, but typically the realization of these patterns in one or more existing approaches is discussed 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 insights into comparative strengths and weaknesses. Moreover, we report on the results of a usability evaluation of the patterns that we conducted with BPM practitioners.

The remainder of this paper is structured as follows. Section II describes and justifies the approach used. Section III presents the various patterns while Section IV evaluates the support offered by mainstream BPM languages and tools for the identified patterns. Section V presents findings from the usability evaluation. Next, Section VI discusses related work and Section VII concludes the paper.

II. Methodology

In this paper we identify *patterns to reduce the perceived model complexity* without changing the abstract syntax, i.e., the goal is to simplify the *representation* of the process model.

The most well-known patterns collection in the IT domain is the set of design patterns documented by Gamma, Helm, Johnson, and Vlissides [20]. This collection describes a set of problems and solutions frequently encountered in object-oriented software design. The success of the patterns described in [20] triggered many patterns initiatives in the IT field, including the Workflow Patterns Initiative [61]. The idea to use a patterns-based approach originates from the work of the architect Christopher Alexander. In [5], he provided rules and diagrams describing methods for constructing buildings. The goal of the patterns documented by Alexander is to provide generic solutions for recurrent problems in architectural design. The idea to use patterns for design problems in the IT domain is appealing as is reflected by the many patterns collections that emerged in the last decade.

The patterns described in this paper have been collected by extensively analyzing existing BPM literature. In addition, we analyzed existing or proposed standards governed by standardization bodies such as OASIS, OMG, W3C, and WfMC. We also analyzed the features of existing tools (business process modeling tools, WFM systems, and BPM systems). Finally, we asked BPM experts and practitioners to comment on the patterns we identified. This resulted in the eight patterns presented in this paper. All operate on the concrete syntax of a process model and aim to reduce the perceived complexity of the model. We could have documented more patterns. However, an important requirement is that a pattern should recur frequently. For each pattern in this paper, we found more than five languages, research approaches or tools that use it.

As it is usual in this domain, we used a fixed pattern-format to document each pattern and used this to evaluate existing languages and tools in a systematic manner. The format lists five elements for each pattern: (a) description, (b) purpose, (c) rationale, (d) realization, and (e) example.

III. Patterns for Concrete Syntax Modification

As a result of our study, we identified eight patterns operating exclusively on the concrete syntax of a process model and classified them according to the hierarchy shown in Figure 1. The first pattern, namely *Layout Guidance*, describes features to modify the process model layout. Four patterns outline visual mechanisms to emphasize certain aspects or parts of a process model (node *Highlight* in Figure 1). These are *Enclosure Highlight*, *Graphical Highlight*, and two annotation patterns: *Pictorial Annotation* and *Textual Annotation*. Two representation patterns, *Explicit Representation* and *Alternative Representation*, refer to the availability of explicit and alternative visual representations for process modeling constructs. The last pattern, *Naming Guidance*, refers to naming conventions or advice to be used in a process model.

In the following we provide a detailed description of each pattern. For illustration purposes, we use the BPMN (Business Process Modeling Notation) standard [44]. An overview of the graphical representation of the main concepts of this notation can be found in Figure 2. Detailed knowledge of this standard is not required to understand the examples in this paper.

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 [37]. Crossing edges negatively affect model understanding [47].

Realization Some languages provide general guidelines on how a model should be laid out on the canvas. eEPCs [26], [15] prescribe to model processes from top to bottom [26], 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” [44, 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 and Pallas Athena Protos, provides limited to no layout support. In the literature, the problem of finding an optimal placement of model elements

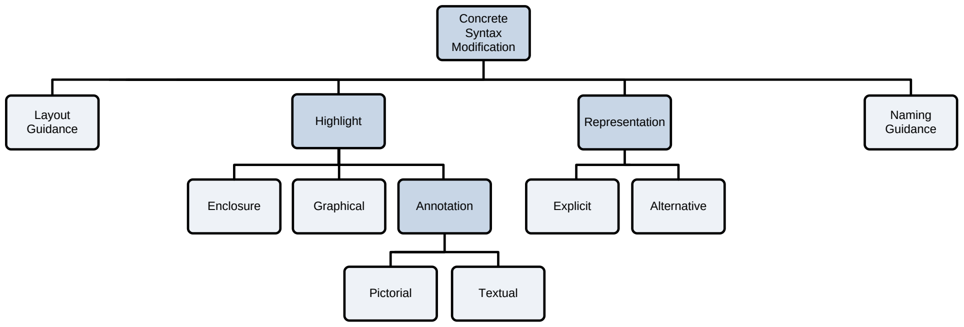

Fig. 1. Patterns for concrete syntax modification.

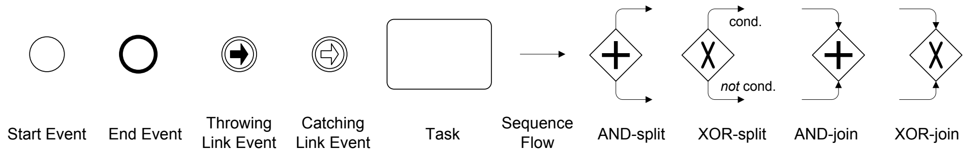

Fig. 2. BPMN 1.2 modeling elements used in this paper.

on the canvas has received quite some attention. Alpfelbacher et al. [7] suggest to place related elements spatially close to each other, Huotari et al. [24] and Purchase [47] point out that crossing edges should be avoided if possible, while Jensen [25] 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 [27], while [18] 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 [53].

Example Figure 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. Figure 3b shows the same model after repositioning the elements according to the BPMN guidelines.

Pattern 2 (*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 [29].

Realization As per languages, BPMN is the only on 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. Enterprise Architect 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 [29] in 1935, 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 Figure 4 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 Figure 3b.

Pattern 3 (*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 [34], [28], [42].

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

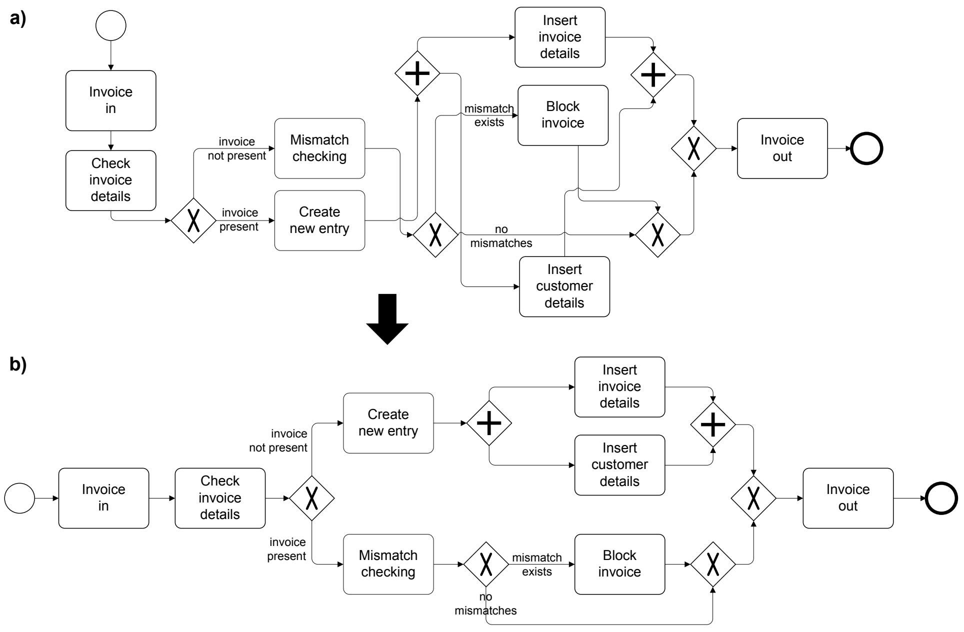

Fig. 3. a) A BPMN model not following any layout guidelines. b) The same model after applying the BPMN layout guidelines.

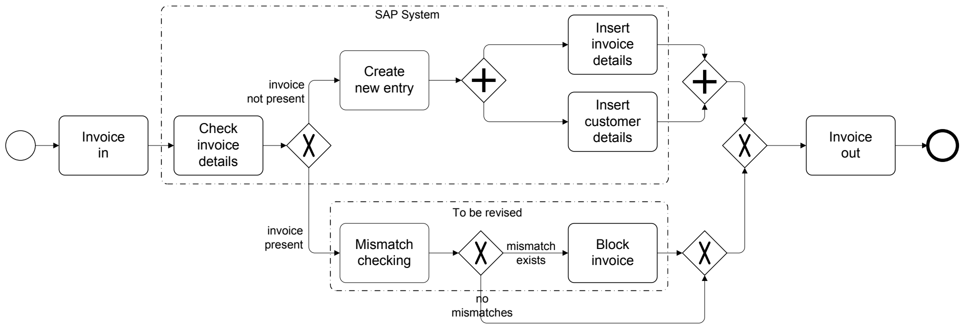

Fig. 4. An example of Enclosure Highlight using the BPMN Grouping construct.

flexibility in elements' size, color and line style, except for specific elements such as throwing and catching events, for which specific guidelines are indicated. Tools that support eEPCs such as Microsoft 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 Enterprise Architect 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 [50], while in [16] the idea of color-coding matching splits and joins is implemented for the WoPeD tool. In [17] a method is presented to represent different types of BPMN elements by objects differing in color and shape; in [22] 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 Figure 5 uses colors to highlight matching splits and joins, and thick edges to highlight the most traversed path, for the model in Figure 3b.

Pattern 4 (*Pictorial Annotation*)

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

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 [45] by speeding up recognition and recall, especially for naive users [42].

Realization In BPMN 2.0 [43], 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, and 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

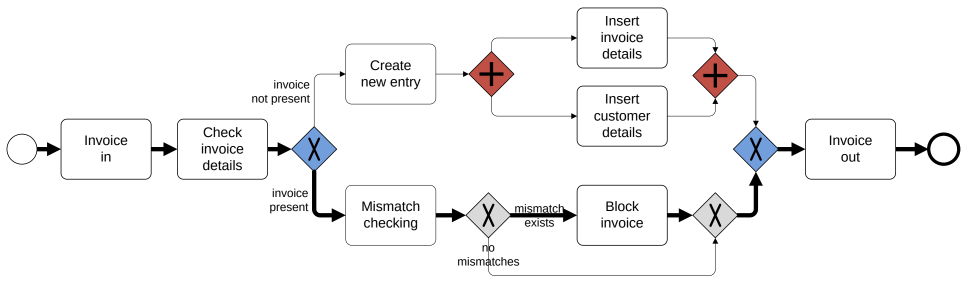

Fig. 5. Two examples of Graphical Highlight: coloring and line thickening.

manual from automated BPMN tasks. In other tools, such as Enterprise Architect and the YAWL Editor, icons or images are fully customizable. For example, in Enterprise Architect one can replace the background of a UML activity with an image. Mendling et al. [36] 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 Figure 6 each task from Figure 3b has been annotated with an icon. For example, task "Block Invoice" features an icon indicating danger while task "Check invoice details" features a lens (reinforcing purpose).

Pattern 5 (*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 either improve understanding of diagrams as comments can improve understanding of source code [42] or augment the model with further information such as time, cost, or quality [54].

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. Enterprise Architect, Intalio|Designer, ARIS and Oryx). Many modeling editors offer proprietary features to visualize free-form text, e.g., ARIS and Protos have text areas while Oryx has Text Notes for eEPCs.

Example The model in Figure 6 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).

Pattern 6 (*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 [34].

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 [23] 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, respectively, 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 covering all modeling concepts (i.e., all notions supported are visualized). However, these two languages provide only few concepts. 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 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). Such a skin however often imposes restrictions based on the underlying model, e.g., the BPMN models need to be block-structured such that each split has a corresponding join of the same type.

Example The models in figures 3-6 are all examples of process models whose modeling concepts (task, gateway, events, sequence flow) are explicitly represented via a dedicated graphical notation.

Pattern 7 (*Alternative 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 potentially reduce model size, especially in large or complex models.

Rationale Specific classes of users may be more familiar with certain symbols [6]. Reducing model size positively affects model understanding [37].

Realization A typical case of alternative representation is that of the OR-split gateway, which replaces a combination of XOR and AND split gateways. This is supported by eEPCs,

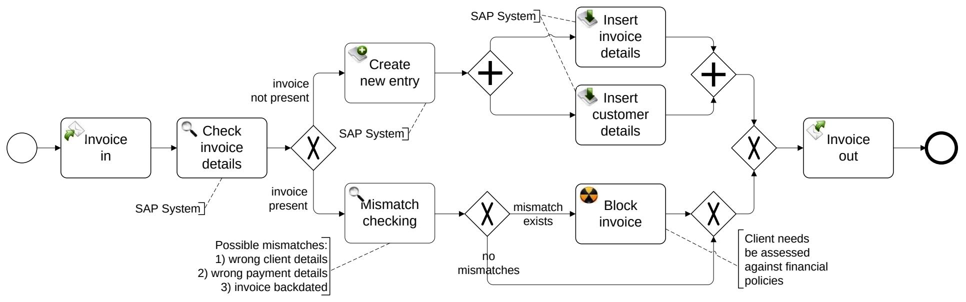

Fig. 6. Examples of Pictorial and Textual Annotations for the model in Figure 8b.

BPMN and YAWL. BPMN also offers a number of alternative (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); a structured loop can be replaced by an activity annotated with a *Loop* marker. UML ADs offer similar alternative notations for the AND-split and the (X)OR split, while the AND-join is subsumed by multiple incoming edges to an activity. Semantically, conditions and tasks need to alternate in YAWL. However conditions can be omitted from the graphical representation of a model when connected only to one preceding and to one subsequent task. These alternative representations are generally supported by the main modeling editors, see e.g. Enterprise Architect for UML ADs, ARIS and Oryx for BPMN, and the YAWL Editor for YAWL. Moreover, Enterprise Architect allow one to replace the predefined shape of a modeling element with an image whereas ARIS provides a Symbol Editor to create alternative representations for each modeling element. These symbols can be organized in templates for specific needs (e.g. a template for presentation to a business audience) and be applied systematically to a process model. In the literature, this feature was already envisaged by Becker et al. [9] under the name of *Representation Variation*, through which the classical EPC symbols can be replaced with custom-made ones, as part of adapting a process model to the requirements of an organization. An occurrence of this pattern is in [17], [18], where the authors define an alternative representation for those edges with numerous bends or that cross other edges: they cut the edge and insert two pointers at the two ends. In this way, clutter can be reduced.

Example Figure 7 shows the model in Figure 3b after applying the BPMN alternative representation for XOR-split, XOR-join and AND-split.

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 [38]. Names that better convey the modeler's intention improve understanding [10].

Realization None of the languages examined provides naming conventions or advice. 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. [38] 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 [39] and in [55]. Silver [55] also proposes naming guidelines for gateways and certain types of events in BPMN 2.0. From a semantic perspective, Becker et al. [10] 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 [51] 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 [19], while in [38] 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. [36] propose a set of 25 frequently occurring verbs of general use, which they extracted from the SAP R/3 reference model [13] and generalized via established verb taxonomies. Tool-wise, renaming features for task and process labels are explored (but not implemented) in [59], as part of a set of refactoring mechanisms for process models. A first effort towards the automation of renaming mechanisms is made in [8], where a prototype implementation is shown that can enforce specific naming conventions for eEPC elements, via thesauri and linguistic grammars. [33] implements a tool to automatically refactor action-noun labels into verb-object labels in process models. The ARIS documentation [15] indicates general semantic guidelines for eEPCs (e.g. avoiding generic verbs such as "to process") while Signavio features a central dictionary of terms which provides auto-completion of labels. However, major tools still neglect the importance of providing automated naming guidance.

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

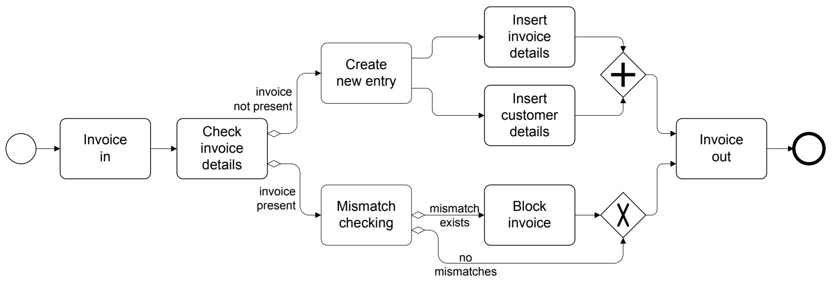

Fig. 7. Alternative Representation of splits and joins for the model in Figure 3b.

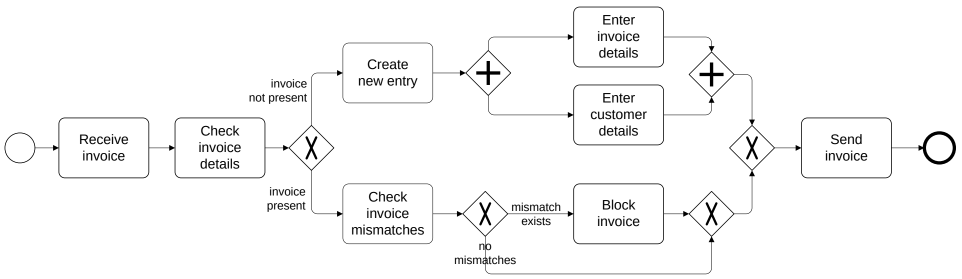

Fig. 8. Renaming the activity labels in Figure 3 according to the verb-object style.

IV. Benchmarking

In this section, 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 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 8.0.2[1]). For each language, we also evaluated at least one supporting modeling editor. For UML ADs 2.1.1 we evaluated Sparx Enterprise Architect 7.1; for eEPCs and BPMN 1.2 we evaluated ARIS 7.1 from Software AG and Oryx 2.0 beta; for BPMN 2.0 we evaluated Oryx 2.0 beta; for BPEL 1.2 Oracle JDeveloper 11.1.1.1.0; for YAWL 2.0 the YAWL Editor 2.0 and for Protos the Protos Editor 8.0.2 from Pallas Athena. Table I 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 because it is vendor specific (although based on Petri nets). 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 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 Alternative Representation, we rated tools with a '+' only if they provide a means to replace standard symbols with custom-made ones. 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.

[1]Protos is now part of the BPM|one suite that allows Protos models to be executed. However, the meaningful enactment of such models requires more modeling efforts, e.g., designing data structures and forms. In most organizations, Protos is mainly used for process modeling and analysis and not for enactment.

		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	Enclosure Highlight	-	+	-	+	-	+	+	+	+	-	-	-	-	+
3	Graphical Highlight	-	+	+	+	+	+	+	+	+	-	+/-	-	+	-
4	Pictorial Annotation	-	+	-	-	-	-	-	-	+	-	+/-	-	+	+/-
5	Textual Annotation	-	+	-	+	+	+	+	+	+	-	-	-	-	+
6	Explicit Representation	+/-	+	+	+	+	+/-	+	+	+/-	-	+	+/-	+	+/-
7	Alternative Representation	+	+	+	+	-	+	+	-	+	-	-	+	-	-
8	Naming Guidance	-	-	-	+/-	-	-	-	-	-	-	-	-	-	-

TABLE I
Evaluation results.

From the table we can make the following observations. First, as expected, the selected tools generally offer wider pattern support than the respective languages. Consider, for example, the differences between UML ADs and Enterprise Architect, 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 fully supports six patterns, while JDeveloper 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 fact that, until recently, such things could not be supported easily. Recent advances in computer graphics make it possible to use pictorial annotations. Moreover, there is a growing need for decorating process models with attributes familiar to business users (e.g., icons). 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, as evidenced by recent publications on the topic.

V. Usability Evaluation

In order to evaluate the patterns from a usability perspective, we turned to the technology acceptance model [14] and its adaptation to conceptual modeling [35]. In essence, this theory postulates that actual usage of an information technology artifact—patterns in the case of this paper—is mainly influenced by the perceptions of potential users regarding usefulness and ease of use. Accordingly, a potential user who perceives a pattern to be *useful* and *easy to use* would be likely to actually *adopt* it.

We conducted a series of focus group sessions with professionals to discuss the patterns. Altogether, 15 process modeling experts participated in these sessions, which took place in Eindhoven and in Berlin. While the five persons in Eindhoven had a consulting background, the ten persons in Berlin worked for vendors of business process modeling tools. On average, the participants had seven years of experience with process modeling. They estimated that they had studied about 300 models in the last 12 months, each having on average 20 activities. Due to this extensive exposure to process modeling, they can be considered to be experts.

We used a questionnaire with seven-point scale items adopted from [35] to measure the participants' perceptions on usefulness and ease of use for each of the patterns. Cronbach's α was used as an *ex post* reliability check. The values determined were 0.92 for usefulness and 0.84 for ease of use, which both point at a high internal consistency of the used question items. The boxplots in figures 9 and 10 display the outcomes of the data analysis for the patterns' usefulness and ease of use respectively. In a boxplot, the median is shown as a horizontal line in a box representing the interval between lower and upper quartile.

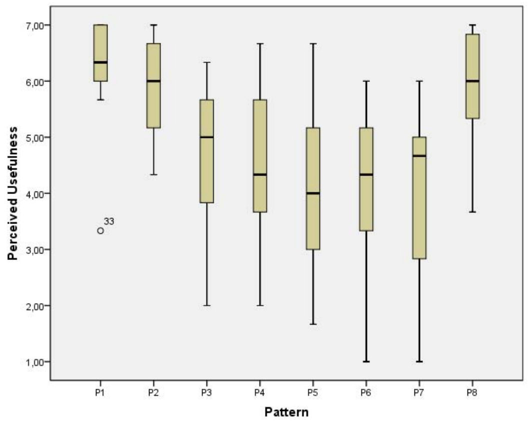

Fig. 9. Perceived usefulness of the patterns.

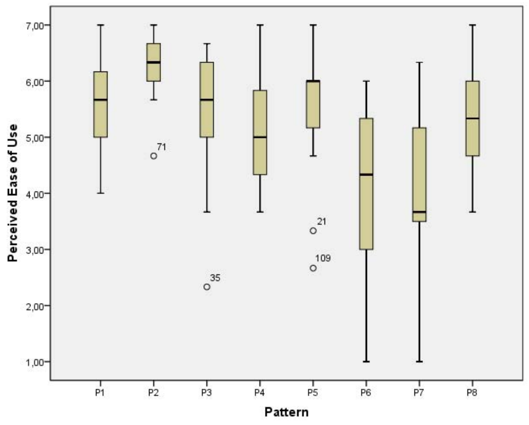

Fig. 10. Perceived ease of use of the patterns.

In Figure 9, it can be seen that all patterns are perceived to be useful (median equals 4 or greater). The patterns that receive the highest scores in this respect are patterns 1, 2, and 8. Figure 10 shows that ease of use is overall considered even more positively with median values of 5 or more for all but patterns 6 and 7. We also recorded statements about the various patterns in a group discussion, which complemented the quantitative assessment. Highly useful patterns were emphasized in this discussion, e.g. a participant from Eindhoven commented on Pattern 2 that it is "very often used". In addition, there were interesting comments revealing a trade-off that seems to exist with respect to individual pattern usage. For instance, an Eindhoven participant remarked with respect to Pattern 5: "I am very much in favor of keeping all models clean. Adding text makes the view very complex and reduces transparency." In a similar vein, a participant from Berlin stated that using this pattern "may lead to semantical problems, since it is easier to create a note than to model correctly" hinting at likely misuse of this pattern. Also, Pattern 3 appears to bear the risk of being used too excessively, since "overstressing [this pattern] leads potentially to unreadable models." While the benefits of Pattern 7 were noted, a participant from Berlin stated that it is "very difficult to explain to beginners. A language should only provide one way to model a scenario."

Altogether, the focus group sessions reveal that industry experts found the patterns to be useful and in general easy to use. Many of them, however, have to be applied in a parsimonious way to live up to their full potential. The experts stressed that novice modelers need additional training to utilize the patterns in an efficient and effective way, and that there is a risk of misuse. Finally, the assessments on ease of use might also reflect the experts' perception of lacking tool support for applying the patterns at this stage. In this regard, there seems to be a need for innovations; in particular for the support of patterns 6 and 7. Indeed, these areas are not yet heavily investigated and require future research.

VI. Related Work

Three main benchmarking frameworks have been used to evaluate process modeling languages: the workflow patterns framework [61], Bunge, Wand and Weber's (BWW) framework [58], and the Semiotic Quality Framework (SEQUAL) [30]. 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 [61]. 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 [11] 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 [52], [48]. 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 [62].

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 [57], and has later been extended to deal specifically with quality of process models [31]. Nonetheless, the authors themselves acknowledge SEQUAL's "disability (sic) to facilitate precise, quantitative evaluations of models" [31, 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 be related to [42], 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.

Our work has also commonalities with cartography. The first geographical maps date back to the 7th Millennium BC. Since then cartographers have improved their skills and techniques to create maps thereby addressing problems such as clearly representing desired traits, eliminating irrelevant details, reducing complexity, and improving understandability. Cartographers use colors to highlight important cities and roads. The thickness of a line representing a road reflects its importance. This corresponds to Pattern 3. Also, annotations are used to point out important things, e.g., a road that is closed during night or a tunnel where users need to pay toll (cf. Patterns 4 and 5).

VII. 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, complemented by an evaluation of state-of-the-art languages and language implementations in terms of these patterns, and an usability test with practitioners. The results of the usability test demonstrate that all identified patterns are indeed perceived as useful and easy of use, while the evaluation of languages and tools shows that there is generally good support for the majority 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 layouting 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 which subsets of patterns can be combined to increase process model understanding. In fact, there might be cases in which the application of particular combinations of patterns may actually decrease the understanding of a process model, e.g. using pictorial annotation with graphical highlight.

Our patterns refer to features that affect the concrete syntax of a process model only, i.e. its visual appearance. The automatic generation of models using process mining techniques [2] provides new insights on the importance of a model's concrete syntax. In fact, process models discovered from event logs typically tend to be overly complex and thus difficult to read [22]. In these cases, it is of utmost importance to simplify the visual representation of such models for the end user. On the other hand, information extracted from event logs can be used to enrich the visual appearance of existing process models (e.g., by highlighting frequent activities and paths). This is yet another application of the patterns identified in this paper.

There are however other equally important complexity reduction features, e.g. structuring a model, modularizing it into sub-processes or abstracting from certain modeling elements, which we left out. Although these features also affect the visual appearance of a process model, they primarily operate on its abstract syntax and as a result, may lead to changes in its appearance. A complementing patterns collection that systematically describes such features is under development.

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. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.

[3] W.M.P. van der Aalst and Kees M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT Press, 2002.

[4] A.A. Abdul, G.K. Tieng 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] C. Alexander. *A Pattern Language: Towns, Building and Construction*. Oxford University Press, 1977.

[6] T. Allweyer. Modellbasiertes Wissensmanagement. *Information Management & Consulting*, 13:37–45, 1998 (in German).

[7] R. Alpfelbacher, A. Knopfel, P. Aschenbrenner, and S. Preetz. Fmc visualization guidelines. http://www.fmc-modeling.org/visualization_guidelines, 2006. Accessed: Nov 2009.

[8] 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.

[9] J. Becker, P. Delfmann, and R. Knackstedt. Adaptive Reference Modeling: Integrating Configurative and Generic Adaptation Techniques for Information Models. In J. Becker and P. Delfmann, editors, *Reference Modeling Conference*, pages 27–58. Springer, 2007.

[10] J. Becker, M. Rosemann, and C. von Uthmann. Guidelines of business process modeling. In *Proc. of BPM*, volume 1806 of *LNCS*, pages 30–49. Springer, 2000.

[11] M. Bunge. *Treatise on Basic Phylosphy Volume 3: Ontology I – The Furniture of the World*. Kluwer Academic Publishers, 1977.

[12] J. Cardoso, J. Mendling, G. Neumann, and H.A. Reijers. A discourse on complexity of process models. In *BPM Workshops*, volume 4103 of *LNCS*, pages 117–128. Springer, 2006.

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

[14] F.D. Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, 13(3):319–340, 1989.

[15] R.B. Davis. *Business Process Modelling with ARIS: A Practical Guide*. Springer, 2001.

[16] A. Eckleder, T. Freytag, J. Mendling, and H.A. Reijers. Realtime detection and coloring of matching operator nodes in workflow nets. In T. Freytag and A. Eckleder, editors, *Algorithms and Tools for Petri Nets*, pages 56–61. CEUR, 2009.

[17] P. Effinger, M. Kaufmann, and M. Siebenhaller. Enhancing visualizations of business processes. In Ioannis G. Tollis and Maurizio Patrignani, editors, *Graph Drawing*, volume 5417 of *LNCS*, pages 437–438. Springer, 2008.

[18] P. Effinger, M. Siebenhaller, and M. Kaufmann. An interactive layout tool for bpmn. *E-Commerce Technology*, 0:399–406, 2009.

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

[20] 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.

[21] V. Gruhn and R. Laue. Reducing the cognitive complexity of business process models. In *IEEE ICCI*, pages 339–345, 2009.

[22] C.W. Günther and W.M.P. van der Aalst. Fuzzy Mining - Adaptive Process Simplification Based on Multi-perspective Metrics. In *Proc. of BPM*, volume 4714 of *LNCS*, pages 328–343. Springer, 2007.

[23] 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*. Springer, 2010.

[24] 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.

[25] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use.* EATCS monographs on Theoretical Computer Science. Springer, 1996.

[26] 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).

[27] 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.

[28] A. Knopfel, B. Grone, and P. Tabeling. *Fundamental Modeling Concepts: Effective Communication of IT Systems*. John Wiley & Sons, 2006.

[29] K. Koffka. *Principles of Gestalt Psychology*. Harcourt, Brace and Co., 1935.

[30] J. Krogstie, O.I. Lindland, and G. Sindre. Defining quality aspects for conceptual models. In *Proc. of ISCO*, pages 216–231, 1995.

[31] 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.

[32] R. Laue and V. Gruhn. *Technologies for Business Information Systems*, chapter Approaches for Business Process Model Complexity Metrics, pages 13–24. Springer, 2007.

[33] H. Leopold, S. Smirnov, and J. Mendling. Refactoring of Process Model Activity Labels. In *Proc. of Natural Language Processing and Information Systems*, LNCS, pages 268–276. Springer, 2010.

[34] G.L. Lohse. A cognitive model for understanding graphical perception. *Human-Computer Interaction*, 8:353–388, 1993.

[35] A. Maes and G. Poels. Evaluating quality of conceptual modelling scripts based on user perceptions. *Data & Knowledge Engineering*, 63(3):701–724, 2007.

[36] 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.

[37] J. Mendling, H.A. Reijers, and J. Cardoso. What makes process models understandable? In *Proc. of BPM*, volume 4714 of *LNCS*, pages 48–63. Springer, 2007.

[38] J. Mendling, H.A. Reijers, and J. Recker. Activity labeling in process modeling: Empirical insights and recommendations. *Information Systems*, 2009.

[39] 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.

[40] J. Mendling and M. Strembeck. Influence Factors of Understanding Business Process Models. In *Proc. of BIS*, LNBIP, pages 142–153. Springer, 2008.

[41] B. Meyer. *Introduction to the Theory of Programming Languages*. Prentice-Hall, Englewood Cliffs, New Jersey, 1990.

[42] 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.

[43] OMG. *Business Process Model and Notation (BPMN), ver. 2.0 beta*, May 2009. http://www.omg.org/cgi-bin/doc?dtc/10-06-04.

[44] OMG. *Business Process Modeling Notation (BPMN),Version 1.2*, January 2009. http://www.omg.org/docs/formal/09-01-03.pdf.

[45] A. Paivio. Dual coding theory: Retrospect and current status. *Canadian Journal of Psychology*, 45(3):255–287, 1991.

[46] M. Petre. Cognitive dimensions beyond the notation. *J. Vis. Lang. Comput.*, 17(4):292–301, 2006.

[47] H.C. Purchase. Which aesthetic has the greatest effect on human understanding? In G. Di Battista, editor, *Graph Drawing*, volume 1353 of *LNCS*, pages 248–261. Springer, 1997.

[48] J. Recker, M. Indulska, M. Rosemann, and P. Green. How Good is BPMN Really? Insights from Theory and Practice. In *Proc. of ECIS*, 2006.

[49] J. Recker, M. zur Muehlen, K. Siau, J. Erickson, and M. Indulska. Measuring method complexity: Uml versus bpmn. In *AMCIS*, pages 1–12. AIS, 2009.

[50] H.A. Reijers, T. Freytag, J. Mendling, and A. Eckleder. Syntax highlighting in business process models. *Decision Support Systems*, 2011 (doi:10.1016/j.dss.2010.12.013).

[51] M. Rosemann. *Process Management: A guide for the design of business processes*, chapter Preparation of process modeling, pages 41–78. Springer, 2003.

[52] 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.

[53] M. Schrepfer, J. Wolf, J. Mendling, and H.A. Reijers. The impact of secondary notation on process model understanding. In A. Persson and J. Stirna, editors, *PoEM*, pages 161–175. IFIP, 2009.

[54] D. Schumm, F. Leymann, and A. Streule. Process viewing patterns. In *Enterprise Distributed Object Computing Conference (EDOC), 2010 14th IEEE International*, pages 89–98. IEEE, 2010.

[55] B. Silver. *BPMN Method & Style*. Cody-Cassidy Press, 2009.

[56] 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.

[57] 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.

[58] Y. Wand and R. Weber. On the Ontological Expressiveness of Information Systems Analysis and Design Grammars. *Journal of Information Systems*, 3:217–237, 1993.

[59] B. Weber and M. Reichert. Refactoring process models in large process repositories. In *CAiSE*, volume 5074 of *LNCS*. Springer, 2008.

[60] M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, 2007.

[61] Workflow Patterns Initiative. Home Page. http://www.workflowpatterns.com. Accessed: February 2010.

[62] B. Wyssusek. On Ontological Foundations of Conceptual Modelling. *Scandinavian Journal of Information Systems*, 18(1):63–80, 2006.

Marcello La Rosa is a Senior Lecturer with the BPM research group at the Queensland University of Technology, Brisbane, Australia. He obtained his PhD in Computer Science with the same research group in 2009. His research interests embrace different topics in the BPM area, such as management of large process model collections, process modeling, configuration and automation. Marcello is a senior trainer for professional education courses on BPM and service-oriented architecture topics.

Arthur H.M. ter Hofstede is a Professor at the Faculty of Science & Technology of Queensland University of Technology in Brisbane, Australia. He is co-leader of the BPM research group in this faculty. He is also a Professor in the Information Systems group at Eindhoven University of Technology in Eindhoven, The Netherlands. His main research interests lie in the area of business process automation. He is involved in both the Workflow Patterns Initiative and the YAWL Initiative.

Petia Wohed is an Associate Professor at the Department of Computer and Systems Sciences at Stockholm University, where she is a member of the Information System Laboratory. Since her graduation in 2000, Wohed's main interest is in the area of business process management, in which she has worked on a series of patterns-based analyses of modern process modeling languages and open-source workflow systems.

Hajo A. Reijers is an Associate Professor in the Information Systems group at Technische Universiteit Eindhoven and an Affiliated Professor with the TIAS/Nimbas Business School of Tilburg University. His research interests cover business process redesign, business process modeling, workflow management technology, and simulation. He published over 75 refereed papers and recently served as the co-chair of the Int. Conference on BPM in 2009.

Jan Mendling is a Junior-Professor with the Institute of Information Systems at Humboldt-Universität zu Berlin, Germany. His research areas include business process management, conceptual modeling and enterprise systems. He has published more than 100 research papers and articles. Jan is member of the editorial board of three international journals. He served as co-chair for the Int. Conference on BPM in 2010 and was the initiator of the BPMN workshop series.

Wil M.P. van der Aalst is Full Professor of Information Systems at the Technische Universiteit Eindhoven and Adjunct Professor at Queensland University of Technology. His research interests include workflow management, process mining, Petri nets, process modeling and analysis. Many of his papers are highly cited (he has an H-index of over 75 as per Google Scholar, making him the Dutch computer scientist with the highest H-index) and his ideas have influenced researchers, software developers and standardization committees on process support.