Information Systems and e-Business Management manuscript No.
(will be inserted by the editor)

Structuredness and its Significance for Correctness of Process Models

Ralf Laue[1], **Jan Mendling**[2]

[1] Chair of Applied Telematics, University of Leipzig, Klostergasse 3, 04109 Leipzig, Germany
(e-mail: `laue@ebus.informatik.uni-leipzig.de`)

[2] Institute of Information Systems, Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany
(e-mail: `jan.mendling@wiwi.hu-berlin.de`)

Key words Business Process Modeling, Structuredness, Correctness, Prediction

The date of receipt and acceptance will be inserted by the editor

Abstract Recent research has shown that business process models from practice suffer from several quality problems. In particular, the correctness of control flow has been analyzed for industry-scale collections of process models revealing that error ratios are surprisingly high. In the past the structuredness property has been discussed as a guideline to avoid errors, first in research on programming, and later also in business process modeling. In this paper we investigate the importance of structuredness for process model correctness from an empirical perspective. We introduce definitions of two metrics that quantify the (un)structuredness of a process model, the Degree of Structuredness and the Unmatched Connector Count. Then, we use the Event-driven Process Chain (EPC) models of the SAP Reference Model for validating the capability of these metrics to predict error probability. Our findings clearly support the importance of structuredness as a design principle for achieving correctness in process models.

1 Introduction

The importance of conceptual models for facilitating a systematic design of information systems has already been recognized in the 1960s (Wand and Weber, 2002). In more recent years, the modeling of *business processes* became more and more important (Davies et al., 2006). Models resulting from such efforts are commonly referred to as *business process models*, or *process models* for short. They are used to support the analysis and design of, for example, process-aware information systems (Dumas et al., 2005), service-oriented architectures (Erl, 2005), and web services (Ferris, 2003).

Process models typically capture in some graphical notation what tasks, events, states, and control flow logic constitute a business process. A business process that is in place to deal with complaints may, for example, contain a task to evaluate the complaint, which is followed by another one specifying that the customer in question is to be contacted. Process models may also contain information regarding the data that is processed by the execution of tasks, which organizational and IT resources are involved, and potentially capture other artifacts such as external stakeholders and performance metrics, see e.g. Scheer (2000). Similar to other conceptual models, process models are first and foremost required to be intuitive and easily understandable, especially in IS project phases that are concerned with requirements documentation and communication (Dehnert and Van der Aalst, 2004).

The quality of such business process models has been discussed for a while, for instance in Guidelines of Modeling (Becker et al., 2000) and the SEQUAL framework (Lindland et al., 1994; Krogstie et al., 2006). On the downside there is a notable gap of empirical research on process model quality aspects (Moody , 2005). There are only few empirical contributions reported that partially help the modeler to come up with a good design. In particular, the works of Gruhn and Laue (2007b), Vanhatalo et al. (2007), and Mendling et al. (2008) show that business process and workflow models from practice have considerable error ratios with 5% to 30% of the models having control flow problems. Structuredness is an important property in this context, requiring that splits have always a matching join, and that these pairs are nested within each other. The prediction function estimated in Mendling et al. (2007a) reveals that unstructuredness is a factor that is connected with a higher probability of errors. What would be informative in this context is strong empirical evidence on the connection between certain styles of modeling and quality aspects. In this way, respective modeling guidelines can be established. Also, good and bad modeling practices can be made part of the process modeling curriculum at universities.

Earlier research into software engineering, e.g. Oulsnam (1982) and Ammarguellat (1992), emphasizes structuredness as a desirable property and proposes techniques for translating unstructured flowcharts into structured ones. However, these techniques are only partially applicable when concurrency is introduced. Situations, where unstructured process models cannot be translated into behavior equivalent structured ones, are discussed in Kiepuszewski et al. (2000). Therefore, the business process modeler has a much higher responsibility to structure the model from the very beginning since an automatic refactoring is not possible in the general case. To put it differently, “the current unstructured style of business process modeling [. . .] leads to similar problems as spaghetti coding” (Holl and Valentin, 2004). In particular, the transformation of process models to executable BPEL that impose certain structuredness constraints becomes a sophisticated problem that cannot be automated in the general case (Mendling et al., 2006; Ouyang et al., 2006; Aalst and Lassen, 2008). Yet, unstructured models can both be adequate for certain business processes, and they can also be correct from a behavioral point of view. Therefore, a modeler should strive for as much structuredness as possible, but might not always be able to avoid unstructured parts.

While the negative impact of unstructuredness on quality of process models is supported by previous research, there is an ongoing debate on how to measure the degree of (un)structuredness. Essentially, there are two approaches to capture it: reduction of structured parts, and identification of unstructured patterns. This paper contributes to the discussion by comparing the *Degree of Structuredness* measure with a new metric called *Unmatched Connector Count*. We use formal notations to define both metrics. Furthermore, we provide an empirical validation based on the SAP reference model. It serves as a sample to evaluate in how far the different metrics are statistically associated with formal control flow errors such as deadlocks. We use Event-driven Process Chains (EPCs) to illustrate our arguments, basically because EPCs cover the typical routing elements that are also used in other languages like YAWL and BPMN, and because we can use existing EPCs from practice (those of the SAP reference model) to test the predictive power of the metrics regarding error probability.

Against this background, the remainder of the paper is structured as follows. In Section 2 we define EPCs and EPC Soundness. Section 3 introduces the two structuredness metrics, the Degree of Structuredness (DoS) and the Unmatched Connector Count (UCC). Subsequently, in Section 4 we calculate these metrics and EPC soundness for each of the EPCs in the SAP reference models. Based on this data, we apply statistical analysis in order to assess the impact of the two metrics on error probability of the EPC models. Section 5 discusses the findings in relation to related work. Finally, Section 6 concludes the paper.

2 Preliminaries

In this section we define EPCs and EPC soundness. Both formalizations are important later in the paper for the definition of structuredness metrics and for understanding the correctness criterion that we use for validating the capability of these metrics to predict error probability.

2.1 Event-driven Process Chains (EPCs)

The Event-driven Process Chain (EPC) is a business process modeling language for the representation of temporal and logical dependencies of activities in a business process (Keller et al., 1992). This language is widely used in the industry, mainly due to the fact that it has been used as the modeling language of the ERP software SAP R/3.

EPCs offer *function type* elements to capture activities of a process and *event type* elements describing pre-conditions and post-conditions of functions. Furthermore, there are three kinds of *connector types* including AND (symbol $\wedge$), OR (symbol $\vee$), and XOR (symbol $\times$) for the definition of complex routing rules. Connectors have either multiple incoming and one outgoing arc (join connectors) or one incoming and multiple outgoing arcs (split connectors). Control flow arcs are used to link these elements. The informal (or intended) semantics of an EPC can be described as follows. The AND-split activates all subsequent branches in concurrency. The XOR-split represents a choice between one of several alternative branches. The OR-split triggers one, two or up to all of multiple branches based on conditions. The AND-join waits for all incoming branches to complete, and then it propagates control to the subsequent EPC element. The XOR-join merges alternative branches. The OR-join synchronizes all active incoming branches. Its behaviour is called non-local since the state of all transitive predecessor nodes has to be considered (Kindler, 2006).

Figure 1 gives the example of an EPC from the SAP reference model (Keller and Teufel, 1998). It describes the functions and the related events that establish the control flow of the *Profit and Cost Planning* process. The process is started when a plan proposal for sales planning is to be created (top right). Depending on whether the sales quantities have to be transferred either the left or the right path is activated, while the event "sales plan is available" will always become true. Therefore, this decision is modeled as an OR-split. Based on different options for new period planning (left top), the cost and activity function is executed. This leads to the product cost planning. When the standard price is updated, the profit planning can be conducted. This eventually contributes to the profit center planning (center bottom).

While the EPC in Figure 1 accurately describes the profit and cost planning, it is complicated by the fact that not all split connectors directly have a matching join connector, i.e. it is not structured. Figure 2 shows three further example EPCs that highlight the structuredness property, and how certain patterns lead to unstructuredness. While the EPC in Figure 2(a) is a structured one, the models in in Figure 2(b) and (c) have one split connector in the left subbranch that is not properly matched by a join connector. However, the unstructuredness in the right EPC is somewhat more severe since there are multiple arcs from the split to unmatched connectors.

In the remainder of this section, we give some definitions that allow us to describe the EPC syntax in a formal way. This way we can later define metrics of structuredness, and in particular the concept of a not properly matched connector.

Definition 1 (Preset and Postset of Nodes) *Let N be a set of nodes and $A \subseteq N \times N$ a binary relation over N defining the arcs. For each node $n \in N$, we define $\bullet n = \{x \in N | (x, n) \in A\}$ as its* preset*, and $n \bullet = \{x \in N | (n, x) \in A\}$ as its* postset*.*

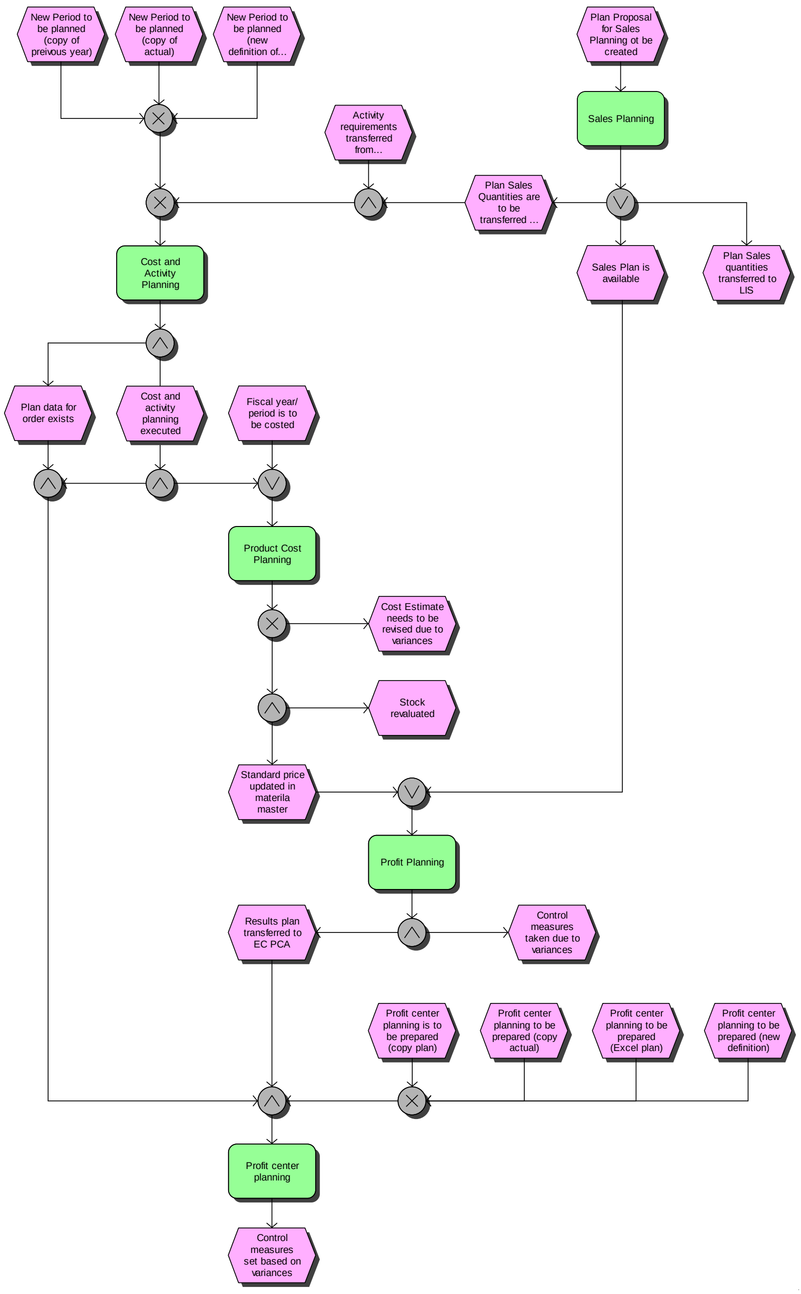
New Period to be planned (copy of preivous year)
New Period to be planned (copy of actual)
New Period to be planned (new definition of...
Plan Proposal for Sales Planning ot be created
Sales Planning
Activity requirements transferred from...
Plan Sales Quantities are to be transferred ...
Cost and Activity Planning
Sales Plan is available
Plan Sales quantities transferred to LIS
Plan data for order exists
Cost and activity planning executed
Fiscal year/ period is to be costed
Product Cost Planning
Cost Estimate needs to be revised due to variances
Stock revaluated
Standard price updated in materila master
Profit Planning
Results plan transferred to EC PCA
Control measures taken due to variances
Profit center planning is to be prepared (copy plan)
Profit center planning to be prepared (copy actual)
Profit center planning to be prepared (Excel plan)
Profit center planning to be prepared (new definition)
Profit center planning
Control measures set based on variances

Figure 1 Profit and cost planning EPC from the SAP reference model

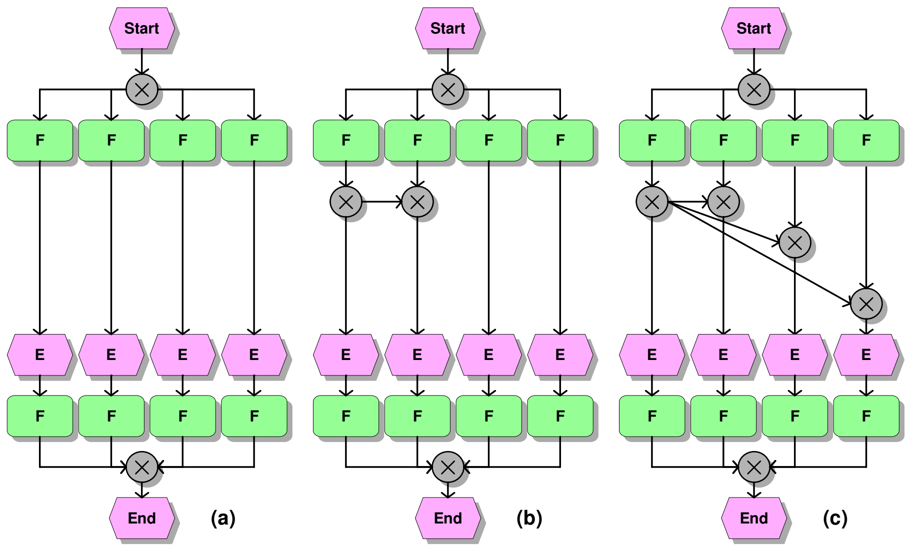

Figure 2 Structured EPC in model (a) and unstructured EPCs in models (b) and (c)

Definition 2 (EPC) *An $EPC = (E, F, C, l, A)$ consists of three pairwise disjoint and finite sets E, F, C, a label function $l : C \to \{and, or, xor\}$, and a binary relation $A \subseteq (E \cup F \cup C) \times (E \cup F \cup C)$ such that*

- *An element of E is called* event. *$E \neq \emptyset$. In particular,*
  $E_s = \{e \in E \mid |\bullet e| = 0 \land |e\bullet| = 1\}$ *is the set of start events and*
  $E_e = \{e \in E \mid |\bullet e| = 1 \land |e\bullet| = 0\}$ *is the set of end events.*
- *An element of F is called* function.
- *An element of C is called* connector. *In particular,*
  $J = \{c \in C \mid |\bullet c| > 1$ *and* $|c\bullet| = 1\}$ *is the set of join-connector and*
  $S = \{c \in C \mid |\bullet c| = 1$ *and* $|c\bullet| > 1\}$ *is the set of split-connectors.*
- *The label l specifies the type of a connector $c \in C$ as* and, or, *or* xor.
- *A defines the control flow. An element of A is called an* arc.
- *An element of the union $N = E \cup F \cup C$ is called a* node.

Definition 3 (Paths) *Let $EPC = (E, F, C, l, A)$ be an EPC and $n_1, n_k \in N$ be two of its nodes. A* path *from node n_1 to node n_k (symbolized by $n_1 \hookrightarrow n_k$) is a sequence of EPC nodes $\langle n_1, n_2, \ldots, n_k \rangle$ such that for all $i = 1, \ldots, k-1$, EPC has an arc from a_i to $a_i + 1$, i.e. $(a_i, a_{i+1}) \in A$. This includes the empty path of length zero, i.e. for any node $a \in N$ there is by definition a path $a \hookrightarrow a$.*

We refer to the set of all nodes on some path between n_1 and n_k as $N_{a \hookrightarrow b} = \{n_1, \ldots, n_k\}$.

Using these definitions of EPCs, preset, postset, and paths we can formalize the minimal structural requirements of an EPC.

Definition 4 (Syntactically Correct EPC) *An $EPC = (E, F, C, l, A)$ is called* syntactically correct *if it fulfills the following requirements:*

1. *EPC is a directed graph such that $\forall n \in N : \exists e_1 \in E_s, e_2 \in E_e$ such that $e_1 \hookrightarrow n \hookrightarrow e_2$.*
2. *There is at least one start event and one end event: $|E_s| \geq 1 \wedge |E_e| \geq 1$.*
3. *Events have at most one incoming and one outgoing arc.* $\forall e \in E : |\bullet e| \leq 1 \wedge |e \bullet| \leq 1$.
4. *Functions have exactly one incoming and one outgoing arc.* $\forall f \in F : |\bullet f| = 1 \wedge |f \bullet| = 1$.
5. *Connectors have at least one incoming and one outgoing arc such that* $\forall c \in C : (|\bullet c| = 1 \wedge |c \bullet| > 1) \vee (|\bullet c| > 1 \wedge |c \bullet| = 1)$.

According to Definition 4 the EPCs of Figure 1 and 2 are syntactically correct.

2.2 EPC Soundness

Soundness is an important correctness criterion for business process models. The original soundness property is defined for a Workflow net (Aalst, 1997), and cannot be used directly for EPCs since the latter may have multiple start and end events. Based on the EPC semantics formalization as a transition system in Mendling and Aalst (2007), we define EPC soundness analogously to soundness of Workflow nets. In this formalization the state of an EPC instance is described based on a so-called *marking* that assigns *tokens* to an arc (symbol: $\sigma(a) = \pm 1$). The work progress of an EPC instance is described as *transitions* (or firings) between different markings. We use the symbol $a \rightarrow b$ for a transition from marking a to marking b and $\overset{*}{\rightarrow}$ for describing the fact that there is a sequence of transitions starting with marking a and ending with marking b. Events and functions simply forward tokens to their output arcs. Connectors process input tokens according to their behavioral semantics, e.g. an XOR-split forwards one input token to one output arc while an AND-join consumes tokens from all input arcs to create one token on its output arc. An EPC starts with an initial marking and terminates with a final marking. The initial marking defines the state of a newly started process by assigning tokens to so-called start arcs. The set of start arcs A_S includes those arcs that point away from start event. A final marking is one for which only incoming arcs of end events are marked. For details see Mendling (2008) and Kindler (2006).

In particular, EPC soundness demands that there is a set of initial markings covering all start nodes, and that for each initial marking in it proper completion is guaranteed. Furthermore, there must exist a set of final markings reachable from some of these initial markings such that there is at least one final marking in which a particular end arc holds a positive token. If that is fulfilled, every arc contributes to properly completing behavior of the EPC. The requirement that the EPC has to be syntactically correct excludes EPCs for which no semantics can be determined, for example, if there are multiple input arcs of a function, or if there are loops without an entry or exit connector.

Definition 5 (Soundness of an EPC) *Let $EPC = (E, F, C, l, A)$ be a syntactically correct EPC, $N = E \cup F \cup C$ its set of nodes, M_{EPC} its marking space, and I_{EPC} and O_{EPC} the set of possible initial and final markings respectively. An EPC is* sound *if there exists a non-empty set of initial markings $I \subseteq I_{EPC}$ and a non-empty set of final markings $O \subseteq O_{EPC}$ such that:*

1. *For each start-arc a_s there exists an initial marking $i \in I$ where the arc (and hence the corresponding start event) holds a token. Formally:* $\forall a_s \in A_s : \exists i \in I : \sigma_i(a_s) = +1$
2. *For every marking m reachable from an initial marking $i \in I$, there exists a firing sequence leading from marking m to a final marking $o \in O$. Formally:* $\forall i \in I : \forall m \in M_{EPC} : \ (i \overset{*}{\rightarrow} m) \Rightarrow \exists o \in O \ (m \overset{*}{\rightarrow} o)$
3. *The final markings $o \in O$ are the only markings reachable from a marking $i \in I$ such that there is no node that can fire. Formally:* $\forall m \in M_{EPC} : ((i \overset{*}{\rightarrow} m) \wedge \nexists m'(m \rightarrow m')) \Rightarrow m \in O$

According to Definition 5 the Profit and Cost Planning EPC of Figure 1 is not sound. Consider the start event *Activity requirements transferred from production planning* at the top in the middle. There is no initial marking including its respective start arc that guarantees proper completion. It might not always get control from the OR-split on the right-hand side. In contrast to that, the EPCs of Figure 2 are sound. Since they have only XOR-connectors, they relate to the Petri net class of state machines that is trivially correct.

3 Structuredness of EPC Business Process Models

There are different characteristics of structuredness, in particular, its extent in relation to the overall model and absolute number of unstructured parts. In this section we describe structuredness in these terms and formalize two metrics: the Degree of Structuredness (DoS) and Unmatched Connector Count (UCC).

3.1 Informal Description of Structuredness

In a well-structured EPC model, splits and joins are properly nested such that each split has a corresponding join of the same type. A pair of splits and joins is properly nested if they are the entry node and the exit node of a single-entry-single-exit component. Such components can be identified using graph parsing techniques Vanhatalo et al. (2007). Structured models can be built iteratively from the building blocks shown in Figure 3. If we look at the splits in such a well-structured model, the following observations can be made:

– For each split s, there is exactly one matching join j.
– For this pair (s, j) the connector labels match, i.e. $l(s) = l(j)$.
– Split s is either the starting split of a structured block or a loop exit.
– If s and j form a loop, $l(s)$ and $l(j)$ must both have an xor label.

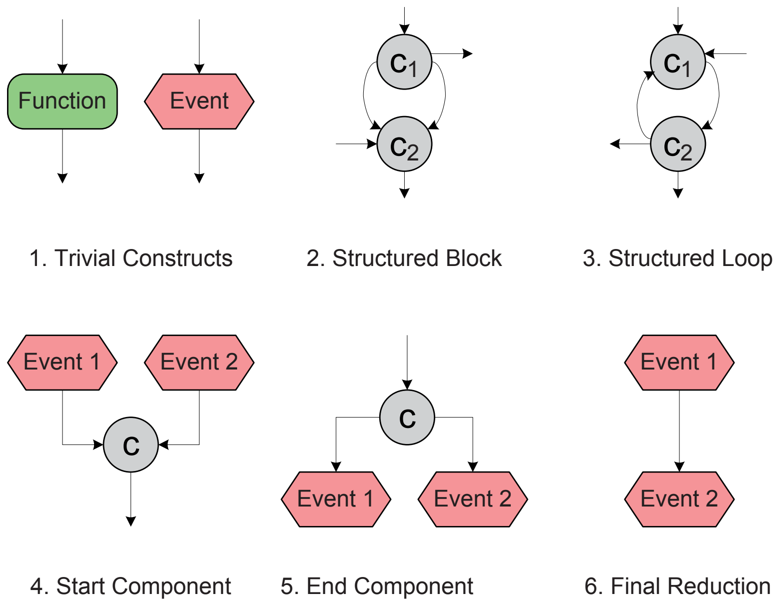

Figure 3 Building Blocks of Structured EPCs (Mendling, 2008)

In the following sections we discuss how the Degree of Structuredness (DoS) and the Unmatched Connector Count (UCC) measure structuredness.

3.2 Degree of Structuredness

In this section we discuss structuredness in terms of a relation between structured and unstructured parts of a process model. Cognitive research into visual programming informs us that 'design is redesign' (Gilmore and Green, 1984): a computer program is not written sequentially; a programmer typically works on different chunks of the problem in an opportunistic order which requires a constant reinspection and comprehension of the current work context. In a structured model, the designer can easily trace the matching pairs of split and join connectors. Therefore, it is quite unlikely that he or she introduces an error in the model. This is difficult if several parts of the model are unstructured. For such parts it is arguably more difficult to understand the control flow. Therefore, errors should be more likely to appear in models with a low degree of structuredness. Measuring structuredness as a degree has obviously some drawbacks. A degree of structuredness will hardly be able to describe how serious the deviations from structuredness are. The fact that the Degree of Structuredness will *decrease* when the model is *enlarged* by adding a sequence of events and functions (without any connectors between them) is a property that is usually regarded as undesired for complexity metrics (Weyuker, 1988; Bache, 1990). Furthermore, there may be a single piece of structure that causes a large part of the model to be unstructured. Clearly, a single metric can only give partial information on a potentially quite complex problem. Still, the degree of structuredness captures one aspect of structuredness that appears important for assessing the design quality of a process model.

Technically, structuredness relates to how far a process model can be built by nesting blocks of matching join and split connectors, see Kiepuszewski et al. (2000). The degree of structuredness can be determined by applying reduction rules and comparing the size of the reduced model to the original size. We specify a *reduction rule T* as a binary relation that transforms a source EPC_1 to a simpler target EPC_2 with less arcs and nodes. In particular, we define the reduction of trivial constructs, structured blocks and loops, and of structured start and end components. In this paper, we provide an abbreviated definition of the rules ignoring arc redirection. A complete definition is given in Mendling (2008).

Definition 6 (Structured Reduction Rules) *Let EPC_1 and EPC_2 be two syntactically correct EPCs, l a label function, N_1, N_2 their sets of nodes, $E_1 \subset N_1, E_2 \subset N_2$ their sets of events, A_1, A_2 their sets of arcs and C_1, C_2 their sets of connectors. The pair (EPC_1, EPC_2) belongs to the binary relation T if one of the following conditions holds:*

1. Trivial Constructs: $\exists n \in N_1 : |\bullet n| = |n\bullet| = 1$.
Construction: $N_2 = N_1 \setminus \{n\}$
2. Structured Blocks: $\exists c_1, c_2 \in C_1 : |\bullet c_1| = |c_2\bullet| = 1 \wedge l(c_1) = l(c_2) \wedge (c_1, c_2) \in A_1$
Construction: $C_2 = C_1 \setminus \{c_1, c_2\}$
3. Structured Loops: $\exists c_1, c_2 \in C_1 : c_1 \neq c_2 \;\wedge\; l(c_1) = l(c_2) = xor \wedge$
$|\bullet c_2| = |c_1\bullet| = 1 \wedge (c_1, c_2) \in A_1 \wedge (c_2, c_1) \in A_1$
Construction: $A_2 = A_1 \setminus \{(c_2, c_1)\}$
4. Structured Start Components: $\exists c \in C_1, e_1, e_2 \in E_1 \wedge$
$|\bullet e_1| = |\bullet e_2| = 0 \wedge a_1, a_2 \in A_1 : e_1 \neq e_2 \;\wedge a_1 = (e_1, c), a_2 = (e_2, c) \in A_1$
Construction: $N_2 = N_1 \setminus \{e_2, c\}$
5. Structured End Components: $\exists c \in C_1, e_1, e_2 \in E_1 \wedge$
$|e_1\bullet| = |e_2\bullet| = 0 \wedge a_1, a_2 \in A_1 : e_1 \neq e_2 \;\wedge a_1 = (c, e_1), a_2 = (c, e_2) \in A_1$
Construction: $N_2 = N_1 \setminus \{e_2, c\}$

6. Final Reduction: $N_1 = \{n_1, n_2\} \wedge A_1 = (n_1, n_2)$
Construction: $N_2 = \emptyset,\ A_2 = \emptyset$

These rules can be applied iteratively until the EPC cannot be reduced any further. We refer to this resulting reduced model as EPC' and define the Degree of Structuredness (DoS) based on its size related to the original size.

Definition 7 (Degree of Structuredness) *Let $EPC = (E, F, C, l, A)$ be a syntactically correct EPC, EPC' the reduced model derived by iteratively applying the reduction rules on EPC, and N_{EPC} and $N_{EPC'}$ their respective sets of nodes. Then, the* degree of structuredness DoS_{EPC} *of the EPC is defined as one minus the number of nodes in EPC' divided by the number of nodes in the original process graph EPC, formally:*

$$DoS_{EPC} = 1 - \frac{|N_{EPC'}|}{|N_{EPC}|}$$

For the Profit and Cost Planning EPC of Figure 1 there are only a few structured parts, i.e. the three and the four start events running into XOR-joins as well as all functions and events with one input and one output arc. Altogether, these are 18 of the 40 nodes. Accordingly, its degree of structuredness is rather low with $18/40 = 0.45$. In the second and the third EPCs of Figure 2 only the sequential functions can be deleted by the reduction rule which reduces the number of nodes by 5, i.e. from 11 to 6 for the left EPC and from 13 to 8 for the right EPC. Based on the definition the Degree of Structuredness is $1 - 6/11 = 0.454$ and $1 - 8/13 = 0.285$, respectively. The first EPC of Figure 2 is completeness structured which is correctly reflected in the metric.

3.3 Unmatched connector count

In this section we discuss structuredness in terms of patterns that cause unstructuredness. The motivation for this discussion again builds on insights from cognitive research (Gilmore and Green, 1984). As mentioned before, in a structured model, the designer can easily trace the matching pairs of split and join connectors. Therefore, it is quite unlikely that he or she introduces an error in the model. In contrast to the degree of structuredness, we might assume that the occurrence of a particular pattern of unstructuredness is a potential source of bad understanding. Accordingly, we deem it worthwhile to consider occurrence of unstructured parts as a unit of measurement. Measuring structuredness as a number of pattern occurrences also has some drawbacks. Most notable, it neglects that deviations from structuredness are much more serious if the model is large. Furthermore, our set of patterns is not complete but builds on a thorough investigation of process models from practice. Yet, we still deem a pattern based approach well suited as it has proven to be appropriate in the related domain of compiler research (Oulsnam, 1982). Of course, and as much as the degree of structuredness, it captures only a single aspect of structuredness, but a one that appears important for assessing the design quality of a process model.

In this section we will define the concepts to measure how much an EPC deviates from structuredness. First, we define cycle entries and cycle exits which we need to formalize the notion of an unmatched connector.

Definition 8 (Cycle Entry and Cycle Exit) *Let $EPC = (E, F, C, l, A)$ be a syntactically correct EPC and $N = E \cup F \cup C$ its set of nodes. Then we define the following notations:*

- *The set of nodes on a directed, non-empty path $N_{a \hookrightarrow a} = \{n_1, \ldots, n_k\}$ with $a = n_1 = n_k$ is called a cycle.*
- *A node $j \in N$ is called cycle entry into a cycle $N_{j \hookrightarrow j}$, if there exist a directed path from a start event to j such that j is the only element of this path that is in the cycle, i.e. if $\exists e_s \in E_s \wedge N_{e_s \hookrightarrow j} : N_{e_s \hookrightarrow j} \cap N_{j \hookrightarrow j} = \{j\}$.*
- *A node $s \in N$ is called cycle exit from a cycle $N_{j \hookrightarrow j}$, if there exists a directed path from s to an end event such that s is the only element of this path that is in the cycle, i.e. if $\exists e_e \in E_e \wedge N_{s \hookrightarrow e_e} : N_{s \hookrightarrow e_e} \cap N_{s \hookrightarrow s} = \{s\}$.*
- *The relation $match(s, j)$ (split s is matched by join j) holds iff there exist two directed paths $P = \langle s, p_1, p_2, \ldots, p_n, j\rangle$ and $Q = \langle s, q_1, q_2, \ldots, q_m, j\rangle$ from s to j whose only common elements are s and j, i.e. $\forall i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, m\} : p_i \neq q_j$.*

Using the above definitions, we can formally count the connectors in an EPC for which one of the structuredness requirements defined in Section 3.1 is violated. Examples for the symptoms for unstructuredness that are considered in the definition are depicted in Figure 4 a) to i). In the following definition we formalize these patterns of unstructuredness. The indices a) to i) directly refer to the figure.

Definition 9 (Not Properly Matched Connectors)
Let $EPC = (E, F, C, l, A)$ be a syntactically correct EPC, $s \in S$ be a split and $j \in J$ a match to s such that $match(s, j)$ is true. We call s not properly matched, *if at least one of the following properties holds:*

(a) There is more than one j for which $match(s, j)$ holds.
(b) For a j with $match(s, j)$, the labels differ: $l(s) \neq l(j)$.
(c) Beyond $match(s, j)$, s is also a cycle exit.
(d) For a j with $match(s, j)$, there is a path from a start node to j which does not pass s or a path from s to an end node which does not pass j.
(e) For a j with $match(s, j)$, there is a cycle $j \hookrightarrow j$ which does not pass s.
(f) The split s is a cycle exit and $l(s) \neq XOR$.
(g) The cycle $s \hookrightarrow s$ has more than one cycle exit.
(h) The cycle $s \hookrightarrow s$ has more than one cycle entry.
(i) There exist two cycles $N^1_{s \hookrightarrow s}$ and $N^2_{s \hookrightarrow s}$, and one cycle entry into $N^1_{s \hookrightarrow s}$ is not a cycle entry into $N^2_{s \hookrightarrow s}$.

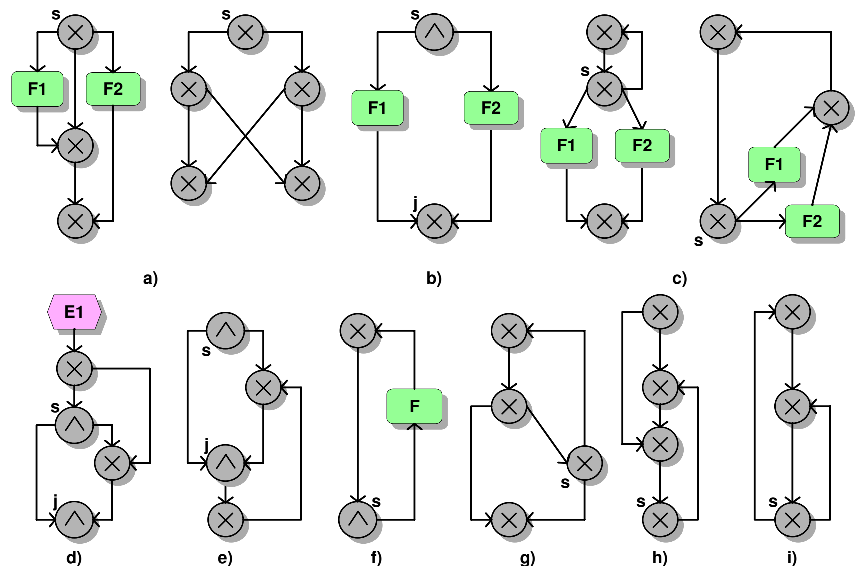

Figure 4 Unstructuredness according to Definition 9

The set of all not properly matched splits is referred to as $S_{\triangle}$.

The idea of our Unmatched Connector Count metric is to count connectors with problems as described in the above definition. However, so far Definition 9 considers only the splits for which no matching join can be found. For both models in Figure 2, only one split that is not properly matched would be counted. However, this would not reflect the fact that the right model has more than one "unstructured arc" originating from this split. In order to consider unmatched joins as well, we make use of the following definition of the inverse of an EPC:

Definition 10 (Inverse EPC) *Let* $EPC = (E, F, C, l, A)$ *be a syntactically correct EPC. Then, we call* $EPC^{-1} = (E, F, C, l, A^{-1})$ *its* inverse, *if* $(y, x) \in A^{-1}$ *if and only if* $(x, y) \in A$.

It is easy to show that the inverse of an EPC is an EPC as well. Splits in EPC become joins in EPC^{-1} and vice versa. Using this definition, we can formalize the Unmatched Connector Count of an EPC as follows:

Definition 11 (Unmatched Connector Count) *Let* $EPC = (E, F, C, l, A)$ *be a syntactically correct EPC, and* EPC^{-1} *its inverse. Then, the* Unmatched Connector Count UCC_{EPC} *is the sum of the number of not properly matched splits of the EPC plus the number of not properly matched splits in the inverse* EPC^{-1}*, formally*

$$UCC_{EPC} = |\{s \in S_{\triangle}\}| + |\{s \in S_{\triangle}^{-1}\}|$$

In the first EPC of Figure 2 there is one mismatched connector in the EPC and one in its inverse. Accordingly, the Unmatched Connector Count is 2. In the second EPC there is also one mismatched connector, but three in its inverse. Thus, its Unmatched Connector Count is 4. In the Profit and Cost Planning EPC of Figure 1 there are altogether 12 unstructuredness patterns.

4 Statistical Prediction of Error Probability

In this section we investigate in how far the two metrics defined in the previous section are capable to predict error probability in EPC process models. An implication of a good predictive power is that a metric can accurately distinguish between models with errors and without errors. We use the EPC soundness criterion as defined in Section 2.2 for determining whether an EPC has errors or not. We assume that a deviation from structuredness is likely to result in errors. Therefore, our hypotheses are:

H1: An decrease in DoS implies an increase in error probability.
H2: An increase in UCC implies an increase in error probability.

We use the EPCs of the *SAP Reference Model* for evaluating these hypotheses. The development of the SAP reference model started in 1992 and first models were presented at CEBIT'93 (Keller and Teufel, 1998, p.VII). Since then, it was developed further until version 4.6 of SAP R/3 which was released in 2000. The SAP reference model includes 604 non-trivial EPCs. The advantage of considering this set of models is that there is extensive literature available that explains its creation, e.g. Keller and Teufel (1998). Furthermore, it is frequently referenced in research papers as a typical reference model and used in previous quantitative analyses reported in Mendling (2008); Mendling et al. (2007a, 2008). This way, our results can be compared to these related works. In a preprocessing step, we checked syntactical correctness according to Definition 4. From the 604 models, 600 are interpretable, i.e. they do not include functions and events with more that one input or output arc (Mendling, 2008, p.145). From these models, we also excluded models that were not strongly connected. Altogether, we consider 540 syntactically correct EPCs in the analysis.

As a first step, we use *correlation analysis.* A significant correlation in this context would be a sign that the metrics are closely related to error probability. Technically, we investigated in how the two metrics are capable to rank non-error and error models. This capability can be estimated using the rank correlation coefficient as defined by Spearman. For both metrics there is a strong and 99% significant correlation which matches the expectation of the hypotheses H1 and H2: for UCC it is +0.777 and for DoS -0.500. Furthermore, there is a significant negative correlation between both metrics of -0.45. This suggests that both measure inversely related aspects, which is indeed the case. Furthermore, this signification correlation supports the assumption that both metrics are strongly connected with error probability.

Table 1 Six Logistic Regression Models

	UCC	**C-UCC**	**DoS**	**C-DoS**	**UCC-DoS**	**C-UCC-DoS**
Constant		-3.673		6.534		-0.054
sign.		0.000		0.000		0.954
DoS			-1.858	-10.063	-4.368	-4.307
sign.			0.000	0.000	0.000	0.000
UCC	0.156	0.865			0.748	0.75
sign.	0.000	0.000			0.000	0.000
Classification	0.217	0.906	0.783	0.859	0.902	0.902
Nagelkerke R^2	0.097	0.707	0.482	0.416	0.835	0.732

In a second step, we use *multivariate logistic regression*, a statistical method that is often used in software measurement (Basili et al., 1996). In this way, we can check whether the metrics are significant factors for explaining error probability, and whether they have the assumed effect on error probability. Technically, this statistics tool estimates the coefficients of a linear combination of input parameters for predicting event versus non-event based on a logistic function. In our case, we predict error versus non-error for the EPCs in the SAP reference model based on the two metrics and a constant. The constant captures whether there is some intrinsic error probability, no matter if the model is structured or not. The general idea of logistic regression is to describe the probability of a binary event by its odds; that is the ratio of event probability divided by non-event probability. In the logistic regression (or logit) model the odds are defined as $logit(p) = ln(\frac{p}{1-p}) = \beta_0 + \sum_{i=1}^{k} \beta_i x_i$ for an observation of k independent input variables (in our case metrics). From this follows that

$$p_i = \frac{e^{\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k}}{1 + e^{\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k}}$$

The parameters β_i are estimated through maximization of the log-likelihood.

The cut value of 0.5 defines whether event or non-event (i.e. in our case whether the EPC is sound) is predicted. $Exp(\beta_k)$ gives the multiplicative change of the odds if the input variable β_k is increased by one unit ($Exp(\beta_k) > 1$ increases and $Exp(\beta_k) < 1$ decreases error probability).

The accuracy of the estimated model is assessed based on the significance level of the estimated coefficients, the percentage of cases that are classified correctly, and the share of the variation that is explained by the regression. This share is typically measured using the Nagelkerke R^2 ranging from 0 to 1. For technical details of logistic regression refer to Hosmer and Lemeshow (2000). For applications in predicting errors in process models see Basili et al. (1996); Mendling (2008); Mendling et al. (2007a, 2008).

Table 1 summarizes the six logistic regression models that we estimated for all combinations of Unmatched Connector Count (UCC), Degree of Structuredness (DoS), and a constant (C). For each of the factors we indicate the estimated coefficient and the significance level. The sign of the coefficient shows if the factor has a positive or negative impact on error probability. A significance level lower than 0.05 demonstrates that the factor is meaningful in the combination with the other factors of the same column. The two rows at the bottom with Classification ratio and Nagelkerke R^2 show how good the explanation of the combination of factors of that column is. Usually, values higher than 0.4 are considered as good. The following conclusions can be drawn.

1. Both metrics perform well in predicting error probability. The univariate regression models for both metrics (UCC, C-UCC, DoS, C-DoS) classify 85.9% and 90.6% of the EPCs correctly (using 0.5 as cut-value) with Nagelkerke R^2 ranging between 0.416 and 0.707 if a constant is included. As mentioned before, values greater than 0.4 already indicate an excellent explanatory power of the regression which are rare in real-world applications.
2. Both metrics complement each other. The multivariate regression models including both metrics (UCC-DoS, C-UCC-DoS) have a much better explanation (an excellent Nagelkerke R^2 of up to 0.835) than each univariate regression model alone. Since the constant is not significant in the multivariate model (significance of 0.954), it can be concluded that both metrics together leave only a little share of variation unexplained.

The fact that both metrics complement each other requires some further comments. Apparently, one metric performs well with aspects related to structuredness where the other has weaknesses, as we have discussed above in the motivations and limitations of each metric. The Degree of Structuredness (DoS) has its strength in quantifying the relative share of the EPC that is structured. Still, when the application of reduction rules stops there may be structured parts left in which the unstructured component is nested. In this regard, it captures a lower bound for relative structuredness. On the other hand, the Unmatched Connector Count (UCC) provides absolute information of unstructuredness which might be difficult to compare across EPCs of different size. Both together perform better than the count metrics in Mendling et al. (2008) and almost as good as the regression model in Mendling et al. (2007a) that requires seven metrics.

The different logistic regression models can also be used to predict the error probability of the Profit and Cost Planning EPC that we introduced in Figure 1. We already mentioned its low degree of structuredness (0.45) and that it has numerous unstructuredness patterns (12). Given these values we can calculate, for instance, the prediction for this EPC according to the UCC regression as $e^{0.156*12}/(1+e^{0.156*12}) = 0.867$. This value is greater 0.5, therefore the regression predicts that this EPC is likely to have errors (which is correct). In contrast to that, the single parameter model DoS predicts no errors (which is wrong for the example). All other regression models give a correct prediction for the example.

5 Related Work

This section discusses related work on business process metrics. In essence, related work can be organized in two categories: conceptual work on process model metrics, partially inspired by software measurement, and experimental work on validating process model metrics. In this section, we focus in particular on metrics that consider overall structural aspects of the process model beyond simple count metrics. For an overview of process model metrics in general refer to Cardoso et al. (2006), Gruhn and Laue (2006a), and Mendling (2008).

The early development of process model metrics is greatly inspired by and based on software quality metrics. The latter aim to help designing computer programs that are less error-prone, easier to comprehend and easier to maintain. A survey of existing software metrics can be found in e.g. Kafura (1985) and Vaishnavi et al. (2007). A number of studies demonstrate the significant correlation of software quality metrics with errors in the software design, for instance Kang and Bieman (1999); Selby and Basili (1991); Shen et al. (1985). In the tradition of this work, there are some works in the 1990s that are mainly rooted in software quality measurement. Daneva et al. (1996) introduce a set of complexity indicators for EPCs including function cohesion, event cohesion and cohesion of a logical connector. From a limited validation with 11 EPCs they conclude that their metrics help to identify error-prone model fragments. Morasca (1999) proposes a set of simple metrics for software specifications designed with Petri-nets. He identifies size, length, structural complexity, and coupling as interesting attributes of a design without striving for an empirical validation. The works by Reijers, Vanderfeesten, et al. introduce different coupling and cohesion metrics for guiding the design of a workflow process (Reijers and Vanderfeesten, 2004; Vanderfeesten et al., 2007). Metrics motivated by cognitive considerations are discussed in Gruhn and Laue (2006b) and Vanderfeesten et al. (2008). Further metrics are proposed by Nissen (1998), Balasubramanian and Gupta (2005), Cardoso (2005, 2006), and Rolón Aguilar et al. (2007), but without directly taking structuredness into account.

Lassen and van der Aalst (2008) propose a complexity metric for workflow nets. They decompose a workflow net into components. Such components are atomic patterns of the workflow nets such as sequence, choice, etc. Each such component is given a component weight that aims to reflect the difficulty to understand this component. After decomposing a workflow net into components, a metric is calculated from the component weights. As unstructured components are penalized by a larger component weight, this metric is well-suited for measuring the structuredness of workflow nets. The authors stress that the component weights are defined in a rather ad-hoc way and serve just as an example based on experience. Experiences from attempts to define weights for reflecting the difficulty of building blocks for software show that finding such weights is a difficult task (Gruhn and Laue, 2007a). Nevertheless, the idea used in Lassen and van der Aalst (2008) is well-established for measuring software structuredness (Fenton and Whitty, 1986).

Mendling et al. take an experimental approach towards process model metrics that is driven by the explanatory power of a metric in an empirical setting. In Mendling et al. (2007a) 28 business process metrics (including size, density, structuredness, coefficient of connectivity, average connector degree, control flow complexity, and others) are tested as error predictors on a set of over 2000 process models from different samples. All metrics, except for density and the maximum degree of a connector, are confirmed to be correlated to error-proneness as expected. Another result of this study is a logistic regression model based on the best seven of these metrics. This regression model is able to classify 90% of the process models correctly. In the research reported in this paper, we needed only 2 metrics to achieve the same classification results. Furthermore, a survey on understandability of process models is reported by Mendling et al. (2007b) in relation to the set of metrics mentioned in the previous study. It is concluded that only five metrics have an expected impact on understandability, of which only two have significant values (density, average connector degree). The resulting linear regression model is able to correctly explain in 45% of the cases. Results from a similar experiment are reported in Mendling and Strembeck (2008).

6 Conclusion and Future Work

In this paper we discussed the importance of structuredness for the correctness of process models. In particular, we defined two metrics that capture different aspects of (un)structuredness. Furthermore, we automatically calculated these metrics for the SAP reference model and used the results for estimating a logistic regression model. The multivariate regression based on both metrics performed almost as good as the regression model in Mendling et al. (2007a) that requires seven metrics.

Our findings strongly support the importance of structuredness for the design quality of process models. The relationship of structuredness and correctness is well investigated from a verification perspective, see e.g. Kiepuszewski et al. (2000); Dehnert and Zimmermann (2005). Some authors use transformation techniques to find behavior-equivalent structured models from unstructured ones, either for verification purposes (Zhao et al., 2006) or for model-driven development (Aalst and Lassen, 2008). Still, such a transformation is not always possible (Kiepuszewski et al., 2000). Therefore, we agree with Aalst (1999): “If possible, non-well-structured constructs should be avoided. If such a construct is really necessary, then the correctness of the construct should be double-checked, because it is a potential source of errors.” The metrics discussed in this paper can be used to check the degree of deviation from this ideal.

Our findings have strong implications for the way process modeling is taught and for guidelines on modeling. We already pointed out that unstructured process models might still be correct. Yet, the fact that they are strongly connected with occurrences of errors suggests that structuredness should be preferred. Apparently, and following the 'design is redesign' principle described in Gilmore and Green (1984), unstructured models are likely to be less maintainable and more difficult to understand. The metrics analyzed in this paper can help designers to assess the quality of their models beyond the notions of formal control flow correctness. Deviations from a high degree of structuredness and a high number of unmatched connector counts can be used as a good hint for considering a rework of the model. In this way, a quantitative support for quality assurance in large process repositories with several hundred models can be provided.

Our work also has some limitations. Currently, we have investigated the significance of our metrics for a large collection of EPC process models. While we would argue that the conclusions from this work are likely to be valid also for other activity-based modeling languages like BPMN or YAWL, a respective validation is still missing. Most of the definitions can be easily tailored to the specifics of these language, but some of their features are not considered here. For instance, the BPMN metamodel is much more complex than the one of EPCs. Therefore, we would assume that modelers have more problems with understanding. An investigation of these questions is part of future research.

As future work, we also want to provide better design support for the modeler in case that the process cannot be modeled in a structured way. Research on structured programming has shown that under some circumstances unstructured elements like internal exits from loops (e.g. using `break`) can improve the quality of programs (Roberts, 1995). To judge about problems that could arise from unstructured parts of a model, we aim to extend the list of patterns reported in Gruhn and Laue (2007c) where unstructuredness is unproblematic. Moreover, we aim to conduct further quantitative analyses on error-probability and understandability beyond Mendling et al. (2007a,b, 2008).

References

W.M.P. van der Aalst. Verification of Workflow Nets. In Pierre Azéma and Gianfranco Balbo, editors, *Application and Theory of Petri Nets 1997*, volume 1248 of *Lecture Notes in Computer Science*, pages 407–426. Springer Verlag, 1997.

W.M.P. van der Aalst. Formalization and Verification of Event-driven Process Chains. *Information and Software Technology*, 41(10):639–650, 1999.

W.M.P. van der Aalst and K.B. Lassen. Translating unstructured workflow processes to readable BPEL: Theory and implementation. *Information and Software Technology*, 50(3):131–159, 2008.

Z. Ammarguellat. A control-flow normalization algorithm and its complexity. *IEEE Trans. Software Eng.*, 18(3):237–251, 1992.

R. Bache. *Graph Theory Models of Software.* PhD thesis, South Bank University, London, 1990.

S. Balasubramanian and M. Gupta. Structural metrics for goal based business process design and evaluation. *Business Process Management Journal*, 11(6):680–694, 2005.

V. Basili, L. Briand, and W. Melo. A validation of object-oriented design metrics as quality indicators. *IEEE Transactions on Software Engineering*, 22(10):751–761, 1996.

J. Becker, M. Rosemann, and C. von Uthmann. Guidelines of Business Process Modeling. In W.M.P. van der Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management. Models, Techniques, and Empirical Studies*, pages 30–49. Springer, Berlin et al., 2000.

J. Cardoso. How to Measure the Control-flow Complexity of Web Processes and Workflows. In L. Fischer, editor, *Workflow Handbook 2005.* Future Strategies, Lighthouse Point, 2005.

J. Cardoso. Process control-flow complexity metric: An empirical validation. In *IEEE International Conference on Services Computing (IEEE SCC 06)*, pages 167–173. IEEE Computer Society, 2006.

J. Cardoso, J. Mendling, G. Neumann, and H. Reijers. A discourse on complexity of process models. In J. Eder and S. Dustdar, editors, *Proceedings of the BPM 2006 Workshops, Workshop on Business Process Design BPI 2006, Lecture Notes in Computer Science Volume 4103*, pages 115–126, September 2006.

M. Daneva, R. Heib, and A.-W. Scheer. Benchmarking Business Process Models. IWi Research Report 136, Institute for Information Systems, University of the Saarland, Germany, 1996.

I. Davies, P. Green, M. Rosemann, M. Indulska, and S. Gallo, "How do practitioners use conceptual modeling in practice?" *Data & Knowledge Engineering*, vol. 58, no. 3, pp. 358–380, 2006.

J. Dehnert and W. M. P. van der Aalst, "Bridging the gap between business models and workflow specifications," *International Journal of Cooperative Information Systems*, vol. 13, no. 3, pp. 289–332, 2004.

J. Dehnert and A. Zimmermann. On the suitability of correctness criteria for business process models. In W.M.P. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, editors, *Business Process Management, 3rd International Conference, BPM 2005, Nancy, France, September 5-8, 2005, Proceedings*, volume 3649 of *Lecture Notes in Computer Science*, pages 386–391, 2005.

M. Dumas, W. M. P. van der Aalst, and A. H. M. ter Hofstede, Eds., *Process Aware Information Systems: Bridging People and Software Through Process Technology.* Hoboken, New Jersey: John Wiley & Sons, 2005.

T. Erl, *Service-oriented Architecture: Concepts, Technology, and Design.* Upple Saddle Revier, New Jersey: Prentice Hall, 2005.

N.E. Fenton and R.W. Whitty Axiomatic approach to software metrication through program decomposition. *Computer Journal*, 29(4):329–339, 1986.

C. Ferris, “What are web services?” *Communications of the ACM*, vol. 46, no. 6, pp. 31–32, 2003.

Gilmore, D.J., Green, T.R.G.: Comprehension and recall of miniature programs. International Journal of Man-Machine Studies **21**(1) (1984) 31–48

V. Gruhn and R. Laue. Complexity metrics for business process models. In *Proceedings of the 9th international conference on business information systems (BIS 2006), vol. 85 of Lecture Notes in Informatics*, 2006a.

V. Gruhn and R. Laue. Adopting the cognitive complexity measure for business process models. In Y. Yao, Z. Shi, Y. Wang, and W. Kinsner, editors, *Proceedings of the Firth IEEE International Conference on Cognitive Informatics, ICCI 2006, July 17-19, Beijing, China*, pages 236–241. IEEE, 2006b.

V. Gruhn and R. Laue. On experiments for measuring cognitive weights for software control structures. In D. Zhang, Y. Wang, and W. Kinsner, editors, *Proceedings of the Six IEEE International Conference on Cognitive Informatics, ICCI 2007, August 6-8, Lake Tahoe, CA, USA*, pages 116–119. IEEE, 2007a.

V. Gruhn and R. Laue. What business process modelers can learn from programmers. *Science of Computer Programming*, 65(1):4–13, 2007b.

V. Gruhn and R. Laue. Good and bad excuses for unstructured business process models. In *Proceedings of 12th European Conference on Pattern Languages of Programs (EuroPLoP 2007)*, 2007c.

A. Holl and G. Valentin. Structured business process modeling (SBPM). In *Information Systems Research in Scandinavia (IRIS 27) (CD-ROM)*, 2004.

D.W. Hosmer and S. Lemeshow. *Applied Logistic Regression.* John Wiley & Sons, 2nd edition, 2000.

D. Kafura. A survey of software metrics. In *ACM ’85: Proceedings of the 1985 ACM annual conference on The range of computing : mid-80’s perspective*, pages 502–506, New York, NY, USA, 1985.

B.-K. Kang and J.M. Bieman. A quantitative framework for software restructuring. *Journal of Software Maintenance*, 11:245–284, 1999.

G. Keller, M. Nüttgens, and A.-W. Scheer. Semantische Prozessmodellierung auf der Grundlage “Ereignisgesteuerter Prozessketten (EPK)”. Heft 89, Institut für Wirtschaftsinformatik, Saarbrücken, Germany, 1992.

G. Keller and T. Teufel. *SAP(R) R/3 Process Oriented Implementation: Iterative Process Prototyping.* Addison-Wesley, 1998.

B. Kiepuszewski, A.H.M. ter Hofstede, and C. Bussler. On structured workflow modelling. In B. Wangler and L. Bergman, editors, *Advanced Information Systems Engineering, 12th International Conference CAiSE 2000, Stockholm, Sweden, June 5-9, 2000, Proceedings*, volume 1789 of *Lecture Notes in Computer Science*, pages 431–445. Springer, 2000.

E. Kindler. On the semantics of EPCs: Resolving the vicious circle. *Data & Knowledge Engineering*, 56(1):23–40, 2006.

J. Krogstie, G. Sindre, and H.D. Jørgensen. Process models representing knowledge for action: a revised quality framework. *European Journal of*

Information Systems, 15(1):91–102, 2006.

K.B. Lassen and W.M.P. van der Aalst. Complexity metrics for Workflow nets. *Information and Software Technology*, 51:610–626, 2008.

O.I. Lindland, G. Sindre, and A. Sølvberg. Understanding quality in conceptual modeling. *IEEE Software*, 11(2):42–49, 1994.

J. Mendling, K.B. Lassen, and U. Zdun. Transformation strategies between block-oriented and graph-oriented process modelling languages. In F. Lehner, H. Nösekabel, and P. Kleinschmidt, editors, *Multikonferenz Wirtschaftsinformatik 2006, XML4BPM Track, Band 2*, pages 297–312, Passau, Germany, February 2006.

J. Mendling. *Metrics for Process Models: Empirical Foundations of Verification, Error Prediction and Guidelines for Correctness.*. Volume 6 of *Lecture Notes in Business Information Processing.* Springer-Verlag, 2008.

J. Mendling and W.M.P. van der Aalst. Formalization and Verification of EPCs with OR-Joins Based on State and Context. In J. Krogstie, A.L. Opdahl, and G. Sindre, editors, *Proceedings of the 19th Conference on Advanced Information Systems Engineering (CAiSE 2007)*, volume 4495 of *Lecture Notes in Computer Science*, pages 439–453, Trondheim, Norway, 2007.

J. Mendling, G. Neumann, and W.M.P. van der Aalst. Understanding the occurrence of errors in process models based on metrics. In R. Meersman and Z. Tari, editors, *OTM Conference 2007, Proceedings, Part I*, volume 4803 of *Lecture Notes in Computer Science*, pages 113–130. Springer, 2007a.

J. Mendling, H.A. Reijers, and J. Cardoso. What makes process models understandable? In G. Alonso, P. Dadam, and M. Rosemann, editors, *Business Process Management, 5th International Conference, BPM 2007, Brisbane, Australia, September 24-28, 2007, Proceedings*, volume 4714 of *Lecture Notes in Computer Science*, pages 48–63, Brisbane, Australia, 2007b.

J. Mendling and M. Strembeck. Influence factors of understanding business process models. In W. Abramowicz and D. Fensel, editors, *Proc. of the 11th International Conference on Business Information Systems (BIS 2008)*, volume 7 of *Lecture Notes in Business Information Processing*, page 142153. Springer-Verlag, 2008.

J. Mendling, H.M.W. Verbeek, B.F. van Dongen, W.M.P. van der Aalst, and G. Neumann. Detection and Prediction of Errors in EPCs of the SAP Reference Model. *Data & Knowledge Engineering*, 64(1):312–329, January 2008.

D.L. Moody. Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. *Data & Knowledge Engineering*, 55(3):243–276, 2005.

S. Morasca. Measuring Attributes of Concurrent Software Specifications in Petri-nets. In *Proceedings of the 6th International Symposium on Software Metrics*, pages 100–110. IEEE Computer Society, 1999.

M.E. Nissen. Redesigning reengineering through measurement-driven inference. *MIS Quarterly*, 22(4):509–534, 1998.

G. Oulsnam. Unravelling unstructured programs. *Computer Journal*, 25 (3):379–387, 1982.

C. Ouyang, M. Dumas, S. Breutel, and A.H.M. ter Hofstede. Translating standard process models to bpel. In E. Dubois and K. Pohl, editors, *Advanced Information Systems Engineering, 18th International Conference, CAiSE 2006, Luxembourg, Luxembourg, June 5-9, 2006, Proceedings*, volume 4001 of *Lecture Notes in Computer Science*, pages 417–432. Springer, 2006.

H.A. Reijers and I.T.P. Vanderfeesten. Cohesion and Coupling Metrics for Workflow Process Design. In J. Desel, B. Pernici, and M. Weske, editors, *International Conference on Business Process Management (BPM 2004)*, volume 3080 of *Lecture Notes in Computer Science*, pages 290–305. Springer-Verlag, Berlin, 2004.

E.S. Roberts. Loop exits and structured programming: reopening the debate. In *SIGCSE '95: Proceedings of the twenty-sixth SIGCSE technical symposium on Computer science education*, pages 268–272, New York, NY, USA, 1995.

E. Rolón Aguilar, F. García, F. Ruiz, and M. Piattini. An exploratory experiment to validate measures for business process models. In *First International Conference on Research Challenges in Information Science (RCIS)*, 2007.

A.-W. Scheer, *ARIS - Business Process Modeling*, 3rd ed. Berlin, Germany: Springer, 2000.

R.W. Selby and V.R. Basili. Analyzing error-prone system structure. *IEEE Transactions on Software Engineering*, 17:141–152, 1991.

V.Y. Shen, T.-J. Yu, S.M. Thebaut, and L.R. Paulsen. Identifying error-prone software. *IEEE Transactions on Software Engineering*, 11:317–324, 1985.

V. K. Vaishnavi, S. Purao, and J. Liegle. Object-oriented product metrics: A generic framework. *Information Sciences*, 177:587–606, January 2007.

I. Vanderfeesten, J. Cardoso, and H.A. Reijers. A weighted coupling metric for business process models. In Johann Eder, Stein L. Tomassen, Andreas Opdahl, Guttorm Sindre, editor, *Proceedings of the CAiSE 2007 Forum*, volume 247 of *CEUR Workshop Proceedings*, pages 41–44, Trondheim, Norway, June 2007.

I. Vanderfeesten, J. Mendling, H. Reijers, W. van der Aalst, and J. Cardoso. On a quest for good process models: The cross-connectivity metric. In *CAiSE 2008, Proceedings*, Lecture Notes in Computer Science, 2008.

J. Vanhatalo, H. Völzer, and F. Leymann. Faster and more focused control-flow analysis for business process models through sese decomposition. In B.J. Krämer, K.-J. Lin, and P. Narasimhan, editors, *Service-Oriented Computing - ICSOC 2007, Fifth International Conference, Vienna, Austria, September 17-20, 2007, Proceedings*, volume 4749 of *Lecture Notes in Computer Science*, pages 43–55. Springer, 2007. ISBN 978-3-540-74973-8.

Y. Wand and R. Weber, "Research Commentary: Information Systems and Conceptual Modeling–A Research Agenda," *Information Systems Research*, vol. 13, no. 4, p. 363, 2002.

E.J. Weyuker. Evaluating Software Complexity Measures. *IEEE Transactions on Software Engineering*, vol. 14, no. 9:1357–1365, 1988.

W. Zhao, R. Hauser, K. Bhattacharya, B.R. Bryant, and F. Cao. Compiling business processes: untangling unstructured loops in irreducible flow graphs. *Int. Journal of Web and Grid Services*, 2(1):68–91, 2006.