Detection and Prediction of Errors in EPCs of the SAP Reference Model

J. Mendling [a,*] H.M.W. Verbeek [b] B.F. van Dongen [b]
W.M.P. van der Aalst [b] G. Neumann [a]

[a] *Vienna University of Economics and Business Administration, Augasse 2-6, 1090 Vienna, Austria*

[b] *Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands*

Abstract

Up to now there is neither data available on how many errors can be expected in process model collections, nor is it understood why errors are introduced. In this article, we provide empirical evidence for these questions based on the *SAP reference model*. This model collection contains about 600 process models expressed as *Event-driven Process Chains* (EPCs). We translated these EPCs into YAWL models, and analyzed them using the verification tool WofYAWL. We discovered that *at least 34 of these EPCs contain errors*. Moreover, we used logistic regression to show that complexity of EPCs has a significant impact on error probability.

Key words: Business Process Management; Verification; Event-driven Process Chains; YAWL; Error Prediction; Logistic Regression

---

* Corresponding author

Email addresses: `jan.mendling@wu-wien.ac.at` (J. Mendling), `h.m.w.verbeek@tue.nl` (H.M.W. Verbeek), `b.f.v.dongen@tue.nl` (B.F. van Dongen), `w.m.p.v.d.aalst@tue.nl` (W.M.P. van der Aalst), `neumann@wu-wien.ac.at` (G. Neumann).

1 Introduction

There has been extensive work on formal foundations of conceptual process modeling and respective languages. However, little quantitative research has been reported on the actual use of conceptual modeling in practice [1]. Moreover, literature typically discusses and analyses languages rather than evaluating enterprise models at a larger scale (i.e., beyond "toy examples"). A fundamental problem in this context is that large enterprise models are in general not accessible for research as they represent valuable company knowledge that enterprises do not want to reveal. In particular, this problem affects research on reference models, i.e., models that capture generic design that is meant to be reused as best practice recommendation in future modeling projects. Accordingly, it is so far neither clear how many errors can be expected in real-life business process models; nor is it clear why modelers introduce errors in process models.

One case of a model that is, at least partially, publicly available is the SAP reference model. It has been described in [2,3] and is referred to in many research papers (see e.g. [4–8]). The SAP reference model was meant to be used as a blueprint for roll-out projects of SAP's ERP system. It reflects Version 4.6 of SAP R/3 which was marketed in 2000. The extensive database of this reference model contains almost 10,000 sub-models, several of them EPC business process models [2,9,3]. Building on recently developed techniques to verify the formal correctness of EPC models as reported in [10], we aim to acquire knowledge about how many formal modeling errors can be expected in a large repository of process models in practice, assuming that the SAP reference model can be regarded as a representative example. We will map all EPCs in the SAP reference model onto YAWL models [11] and use the WofYAWL tool [10] as a means to verify their correctness using the relaxed-soundness criterion [12,13]. In a relaxed sound process there is a proper execution sequence for every element, but a proper completion is not guaranteed. We have to stress that this analysis yields a lower bound for errors since there are process models that are relaxed-sound but not correct against the more restrictive soundness criterion [14]. To be more concise, our analysis covers only formal control flow errors that affect relaxed soundness. Beyond verification of formal correctness, a process model must also be validated to make sure that all real-world scenarios are handled as expected [15]. Since WofYAWL cannot check whether real-world processes are modeled appropriately, validation is not subject of our analysis. As a consequence, it has to be expected that there are more errors than those that we actually identify using the WofYAWL verification approach.

It is a fundamental insight of software engineering that errors should be detected as early as possible in order to minimize development cost (see e.g. [16,17]). Therefore, it is important to understand why and in which circumstances errors occur. Several research in software engineering was conducted on complexity metrics as determinants for errors (see e.g. [18–22]). A similar hypothesis that complexity is a driver for errors has recently be formulated in [23] in the context of business process modeling. Yet, there is no evidence to support it. Even measuring complexity of business processes is still too little understood. We will use the sample of the 604 EPC business process models of the SAP reference model to test whether errors in terms of relaxed-soundness can be statistically explained by complexity metrics.

The remainder of this article is organized as follows. Section 2 describes the design of our quantitative study. In particular, we discuss the mapping of EPCs from the SAP reference model to YAWL models, the analysis techniques employed by WofYAWL, and the identification of how the models can be corrected. In Section 3 we focus on the analysis of the EPCs in the SAP reference model. First, we calculate descriptive statistics that allow us to get a comprehensive inventory of errors in the SAP reference model. Secondly, we investigate the hypothesis that more complicated models have more errors. This hypothesis was suggested in [23], and we analyze it using different complexity measures and by testing whether they are able to explain the variance of errors, i.e. how errors are distributed across EPCs with different measures. The results allow us to conclude which complexity metrics are well suited to explain error variance and that the impact of complexity on error probability is significant. Subsequently, we discuss our findings in the light of related research (Section 4) and conclude with a summary of our contribution and its limitations (Section 5).

2 Detection of Errors in EPCs

In this section, we present the way we evaluated the SAP reference model. In Section 2.1, we start with an introduction to EPCs by the help of an example that we also use to illustrate the verification. As an input for the different analysis steps, we use the ARIS[1] XML export of the reference model (see Fig. 1). In a first step, the EPC to YAWL transformation program generates a YAWL XML file for each EPC in the reference model (see Section 2.2). These YAWL models are then analyzed with WofYAWL that produces an XML error report highlighting the design flaws that have been discovered (see Section 2.3). Independent from these steps, the Model Analyzer extracts descriptive information such as the number of elements of a certain element type and whether there are cycles for each EPC model. An XML file of these model characteristics is then merged with the output of WofYAWL based on the ID of each EPC, and written to an analysis table in HTML format. Then, this table is imported in the software package SPSS to do the statistical analysis. Additionally, Section 2.4 reports on how erroneous EPC models can be corrected.

[1] ARIS Toolset is the commercial business process modeling tool of IDS Scheer AG.

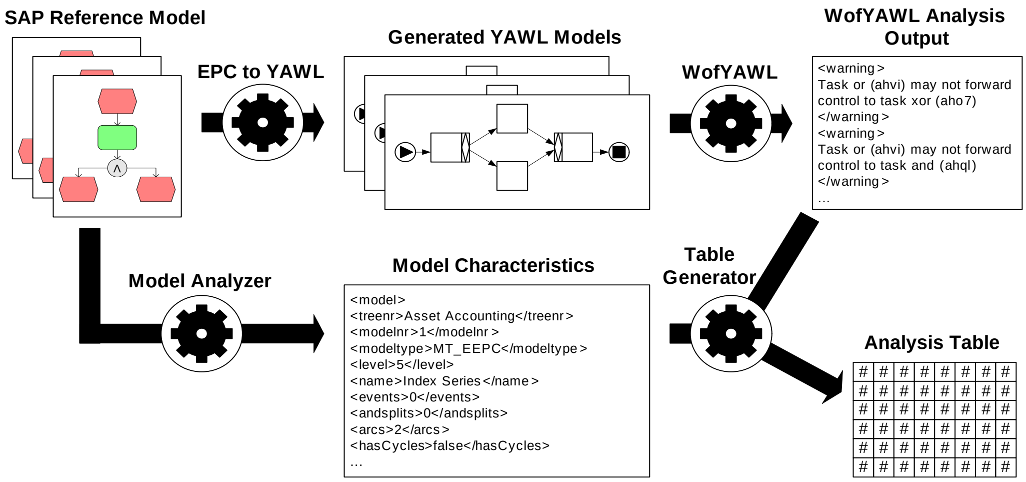

Fig. 1. Overview of the Evaluation Design

2.1 Introduction to EPCs

Event-driven Process Chains (EPCs) are frequently used in large scale modeling projects in practice. In the SAP reference models, EPCs model the business processes which are supported by the SAP system. Fig. 2 shows the EPC model for "Certificate Creation" as an example of one of these models. It is taken from the quality management branch of the SAP model and documents when and how a quality management certificate is created by the help of an information system. The EPC contains three different types of elements: functions, events, and connectors.

Function type elements capture activities of a process (rounded boxes). In the EPC there are three functions capturing the "Certificate Profile and Profile Assignment", "Creation of a Quality Certificate", and "Edit Recipient of Quality Certificate" activities.

Event type elements describe pre- and post-conditions of functions (as hexagons). Accordingly, the EPC model for "Certificate Creation" in Fig. 2 illustrates the temporal and logical dependencies between the three functions by giving their various pre-conditions and post-conditions as events. For example, the "Certificate Profile and Profile Assignment" function results in the event "Certificate profile assignment exists" to be true as a post-condition. This

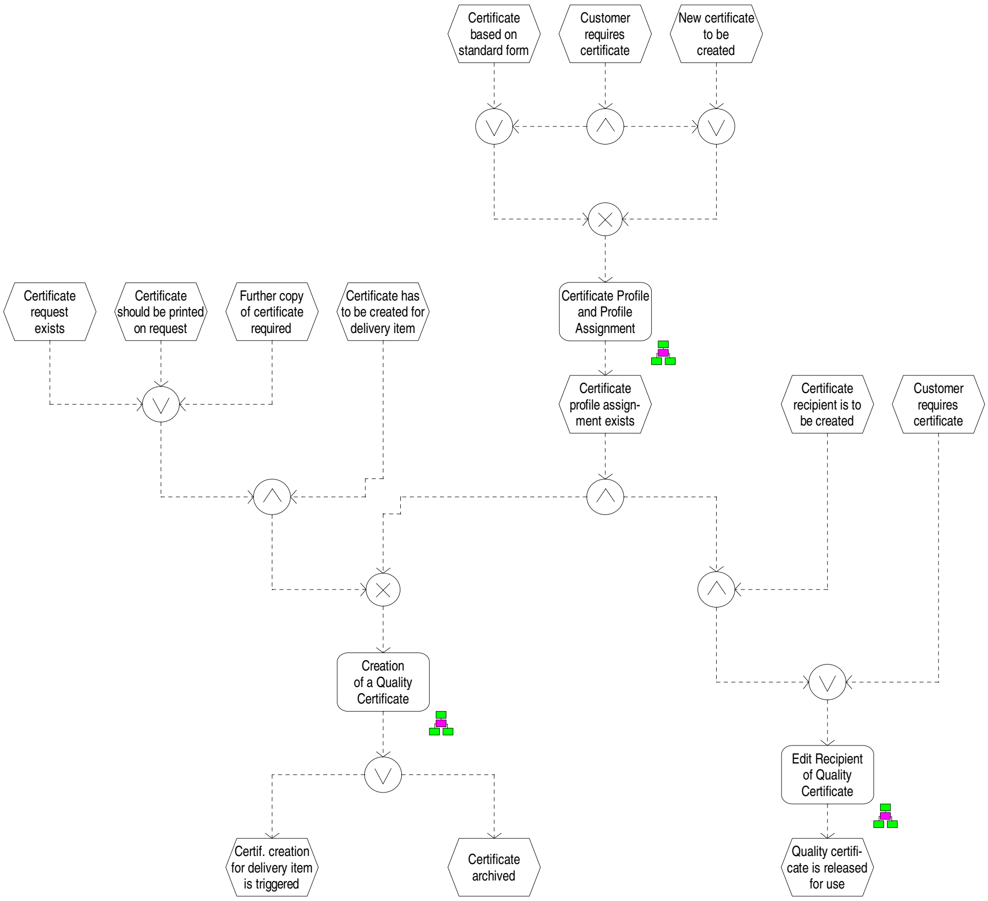

Fig. 2. One of the EPCs in the SAP reference model: the "Certificate Creation" process

event serves as one of the pre-conditions for the "Edit Recipient of Quality Certificate" to be executed. Furthermore, there are three kinds of **connector types** including AND, OR, and XOR 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). The informal semantics of an EPC can be described as follows. The AND-split activates all subsequent branches in concurrency. The XOR-split represents a choice among several alternative branches, i.e., precisely one branch is selected. The OR-split triggers one, two or up to all of the branches, i.e., for each branch a condition is evaluated and depending on the result this branch is taken. In both cases of the XOR- and OR-split, the activation conditions are given in events subsequent to the connector. Accordingly, splits after events followed by multiple functions are forbidden with XOR and OR as the activation conditions do not become clear in the model. The AND-join waits for all incoming branches to complete, after which it propagates control to the subsequent EPC element. The XOR-join merges alternative branches. The OR-join synchronizes all active incoming branches. This feature is called non-locality since the state of all transitive predecessor nodes has to be considered (see e.g. [24]). It poses a major verification challenge since standard Petri nets analysis techniques are not directly applicable. For a formalization of EPC semantics the reader is referred to [24].

2.2 Transformations of EPCs to YAWL

Several mappings from EPCs to Petri Nets have been proposed in order to verify formal properties, see e.g. [24] for an overview. In this paper, we use a transformation from EPCs to YAWL that has been recently defined in [25]. The advantage is that each EPC element can be directly mapped to a respective YAWL element without changing the behavior (see Fig. 3). Furthermore, we can use YAWL verification tools to analyze EPCs. Even though EPCs and YAWL are very similar in terms of routing elements, there are three differences that have to be considered in the transformation: (1) state representation, (2) connector chains, and (3) multiple start and end events.

EPC functions can be mapped to YAWL tasks following mapping rule (a) of Fig. 3. The first difference between EPCs and YAWL is related to *state representation*. EPC events can be interpreted as states that define pre-conditions for the start of functions and post-conditions after their completion. Though this definition might suggest a direct mapping of events to YAWL conditions (the YAWL equivalent to places in Petri nets), things are a bit more complicated. In an EPC it is syntactically correct to model *one event* followed by an

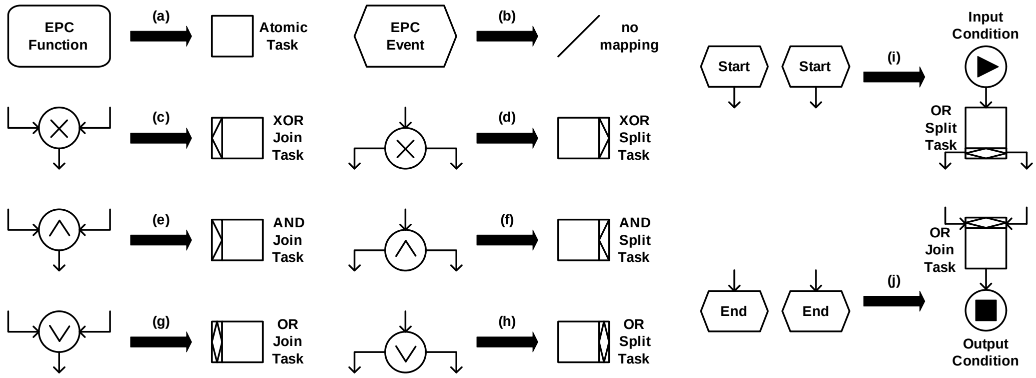

Fig. 3. Overview of the EPC to YAWL Mapping

AND-connector that splits control flow to two functions. In YAWL there are actually *two conditions* required as pre-conditions for the two functions. Accordingly, EPC events are related to states, but there is not a direct one-to-one correspondence between events in EPCs and conditions in YAWL. Therefore, rule (b) in Fig. 3 defines that events are not mapped to YAWL taking advantage of the fact that arcs in YAWL represent implicit conditions if they connect two tasks. Please note that this mapping does not have any impact on the routing between different functions. In EPCs connectors are independent elements. Therefore, it is allowed to build so-called *connector chains*, i.e. paths of two or more consecutive connectors (cf. Fig. 2). In YAWL there are no connector chains since splits and joins are part of tasks. The mapping rules (c) to (h) map every connector to a dummy task with the matching join or split condition (see Fig. 3). The third difference stems from *multiple start and end events*. An EPC is allowed to have more than one start event. Multiple end events represent implicit termination: the triggering of an end event does not terminate the process as long as there is another path still active. In YAWL there must be exactly one start condition and one end condition. Therefore, the mapping rules (i) and (j) generate an OR split for multiple starts and an OR-join for multiple ends. This implies that any combination of start and end events is considered to be correct even if only a restricted set of combinations is meaningful. By using such an interpretation, this mapping yields a YAWL model that includes all execution paths that can be taken in the EPC. We will exploit this later when using the relaxed soundness criterion.

Fig. 4 gives the result of applying the transformation to the "Certificate Creation" EPC of the first section. Note that connectors are mapped onto dummy tasks. To identify these tasks they are given a unique label extracted from the internal representation of the EPC, e.g., task "and (c8z0)" corresponds to the AND-split connector following event "Customer requires certificate".

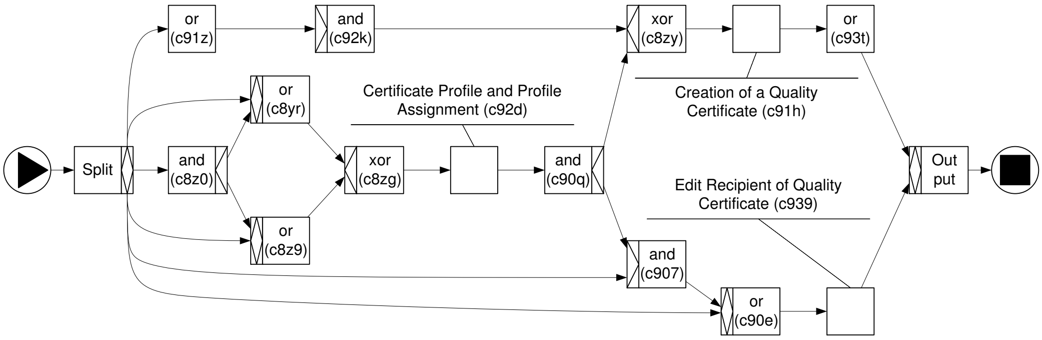

Fig. 4. YAWL model obtained by applying the mapping shown in Fig. 3 to the running example

2.3 WofYAWL Analysis

After mapping the EPC onto YAWL, we can use the verification tool WofYAWL [10]. WofYAWL internally uses a Petri-net representation [2] of the YAWL model for the analysis and translates the result back to warnings that relate to the elements of the YAWL net. As indicated before, we use a correctness criterion based on *relaxed soundness* [12,13]. Relaxed soundness is a "weaker version" of the classical *soundness* notion defined for workflow nets [29,30]. Workflow nets are Petri nets with a single source place (i.e., the start of the process) and a single sink place (i.e., the end of the process). A workflow net is sound if any token put into the source place finally results in a token in the sink place. More precise: From any state reachable from this initial state it is possible to reach the desired end state. Note that soundness excludes deadlocks (the process gets stuck and nothing can happen) and livelocks (the process is trapped in a loop it cannot escape from). Clearly soundness is de-

[2] For details on Petri nets refer to [26–28]

sirable. However, EPCs are frequently used in such a way that the expected behavior is captured, but exceptional situations are not explicitly handled. In such a setting the EPC should be able to terminate properly, but it is not necessary to terminate properly in all possible paths. Therefore, we use relaxed soundness since it precisely matches this requirement. In particular, relaxed soundness demands that any transition (i.e., a task or function) is involved in at least one "sound execution", i.e., for any transition there should be an execution path moving the process from the initial state (one token in the source place) to the desired final state (one token in the output place) [12,13].

As a first step of the relaxed soundness analysis, WofYAWL maps a YAWL model onto a Petri net using the mapping defined in [13]. Fig. 5 sketches a small fragment of the Petri net that results from a translation of the YAWL model shown in Fig. 4. The fragment only considers the dummy tasks resulting from the mapping of the top four connectors in Fig. 2. Moreover, from the initial OR-split task "Split" in Fig. 4 we only consider the arcs connected to these four dummy tasks. Note that when mapping this OR-split onto transitions all possible interpretations are generated ($2^3 - 1 = 7$ transitions). Similarly, all other XOR/OR-splits/joins are unfolded.

Using the notion of relaxed soundness we can label elements of the Petri net using "happy smileys" and "sad smileys". The "happy smileys" are used to identify net elements that are involved in so-called "good execution paths", that is, the execution paths in the Petri net that lead from the initial state to the *desired* final state. Consider for example Fig. 5. In this Petri net there are

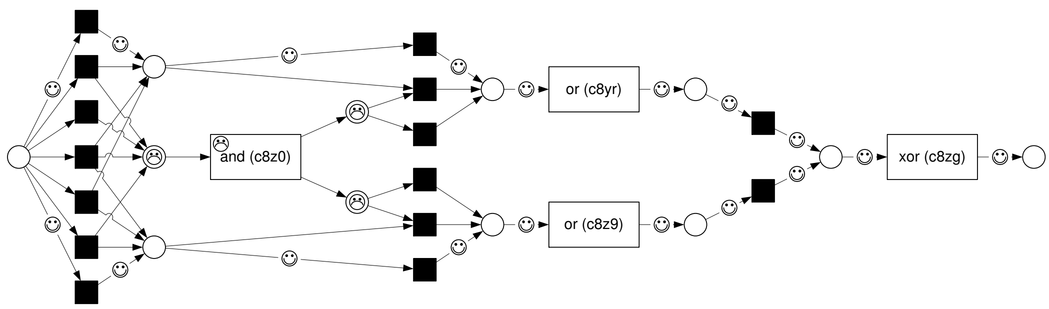

Fig. 5. Petri net fragment of the converted YAWL model

two “good execution paths” which join at the XOR-join named “xor (c8zg)”: the first path passed two black transitions at the very top of the model, reaches the OR-join (c8yr), and arrives at the XOR-join (c8zg) via another black transition. The second path passes the transitions at the bottom of the model including the OR-join (c8z9). The “sad smileys” visualize relevant parts in the Petri net that are not covered by some good execution path: if only the place before the AND has a token, a firing produces one token token on each of the output places of the AND. These can be propagated in such a way that the end place receives one of these tokens while the other one is still in the net. If additionally one of the places below or above the AND input place have a token, they can synchronize with the respective tokens at the AND output places. But here as well, the path at the top and the path at the bottom are also not synchronized. Accordingly, there are in any execution path involving the AND two tokens that reach the XOR. As a result, the AND can in no way contribute to reaching the desired final state from the initial state. WofYAWL issues the following warnings for this fragment:

- `Task "or (c8yr)" may not receive control from task "and (c8z0)"`,
- `Task "or (c8z9)" may not receive control from task "and (c8z0)"`,
- `Task "or (c8yr)" may be an XOR-join instead of an OR-join`,
- `Task "or (c8z9)" may be an XOR-join instead of an OR-join`.

These warnings indicate that there is a problem involving the top four connectors in Fig. 2. Since the AND-split connector splits the flow into two paths that join with an XOR-join, these two paths cannot be involved in a good execution as indicated by first two warnings. Moreover, if the AND-split connector is not allowed to occur, the two OR-joins could as well be XOR-joins. In Section 2.4 we will show how these diagnostics can be used to repair the problem.

In the analysis we use *transition invariants* to avoid constructing large or even infinite state spaces [10]. However, the mapping shown in Fig. 3 tends to generate very large models. For example, in the SAP reference model there are EPCs with 22 end events. Using the naive translation shown in Fig. 3 this results into 4 million transitions just to capture the final OR-join. Therefore, we have used a more refined mapping which scales much better. Moreover, we have used soundness-preserving reduction rules [27] to further reduce the complexity of the models without losing any information. For additional details on this approach, we refer to [10].

2.4 Implications of Errors

Errors in EPCs can be identified in an automated way using WofYAWL. However, being able to detect problems is not enough. In practice, these problems should be repaired by the process owner. While WofYAWL points to the elements causing the problem, there are often several choices for correcting the errors, and the process owner has to identify the solution that matches the desired behavior. Take for example the EPC of Fig. 2. In Section 2.3, we have shown that there were four error messages coming from WofYAWL. From this, it is rather trivial to conclude that the XOR-join does not match the preceding connectors. To repair this mistake, the process owner should decide whether to change the AND-split into an XOR-split, or to change the XOR-join into an AND-split. The decision cannot be made without explicit domain-knowledge of the process under consideration, and might even be different for each implementation of the process. Furthermore, in this example WofYAWL generated a message suggesting that an OR-connector could be changed to an XOR. If such a message is generated for a connector in isolation (i.e. there are no other messages regarding the same connector), then this connector can indeed be changed without disturbing the model. However, if other messages relate to the same connector (which is the case in our example) special care has to be taken. In the "Certificate Creation" model for example, the connectors can only be changed to an XOR-join under the assumption that the event "Customer requires certificate" cannot occur. Since this is not a valid assumption, we propose to repair the EPC as shown in Fig. 6. Figure 7 shows another

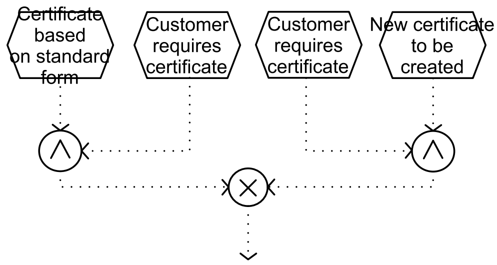

Fig. 6. Fragment of an alternative “Certificate Creation” EPC addressing the problems identified using WofYAWL

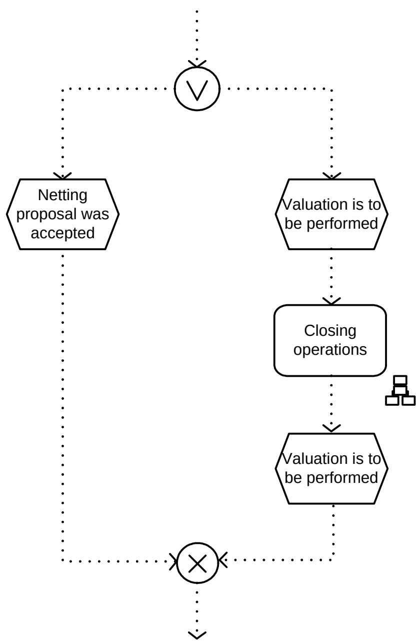

Fig. 7. Fragment of the “Stocks [TR-SE]” EPC

example of an error found by WofYAWL. The EPC is taken from the treasury branch of the SAP reference model. In this model there are basically two choices to cure the problem: either make the OR-split an XOR, or make the XOR-join an OR. WofYAWL proposes the first option, and now, since no other message relates to the mismatch connectors, it is safe to follow this proposal.

3 Prediction of Errors in the SAP Reference Model

Using the approach depicted in Fig. 1 we analyze the SAP reference model. First of all, we locate the parts of the reference model where errors occur most frequently (Section 3.1). Second, in Section 3.2, we formulate hypotheses relating correctness to properties of the EPC (e.g., larger models are more likely to contain errors). Finally, we test these hypotheses using logistic regression (Section 3.3).

3.1 Descriptive Statistics

The sample of the SAP reference model that was available for this research is organized in two orthogonal dimensions: hierarchy levels and branches. Table 1 illustrates that five levels of abstraction are used to arrange the models. Each model at a lower level is a sub-model of a model on a higher level. On the top level there is one model which serves as the root for the model hierarchy. Most of the 9844 models are of model type extended EPC ("eEPC"), but only a fraction of them represent proper EPCs with at least one start event and one function. There are 604 of such process models as listed in the column "EPC". These EPCs have been the starting point of our analysis. Using the transformations and the WofYAWL tool described in Section 2, we discovered that at least 34 models have errors (5.6% of 604 analyzed EPCs).

Table 2 summarizes the SAP reference model subdivided into its 29 branches. It can be seen that the number of EPC models varies substantially (from none in Position Management to 76 in Sales & Distribution). Furthermore, the EPCs are of different size indicated by the mean number of events, functions, connectors, and arcs in columns $E_{av.}$, $F_{av.}$, $C_{av.}$, $A_{av.}$, respectively. The column "Cycle" states how many EPCs have cycles, and "Error" for how many models WofYAWL reports an error. It is interesting to note that branches with more than 10% of faulty models tend to be larger. For example, refer to the Real

Table 1
Hierarchy Levels of the SAP Reference Model

Hierarchy Level	Models	eEPC	Function Allocation Diagram	Process Selection Diagram	Role Activity Diagram	EPC	Error
1	1	1	0	0	0	0	0
2	58	29	0	29	0	0	0
3	175	73	0	0	0	102	15
4	1226	724	0	0	0	502	19
5	8384	3035	3035	0	2014	0	0
All Levels	9844	3862	3035	29	2014	604	34

Estate Management branch: 16.7% of the EPCs have errors and the mean number of events (12.7) per EPC is higher than the overall mean number of events (11.5). Similar observations can be made for functions (6.5 to 4.0), connectors (7.3 to 5.2), and arcs (27.0 to 20.8). In the following subsection, we test whether such characteristics of an EPC can be used to predict errors.

3.2 Hypotheses and Related Error Determinants

Determinants of errors in EPCs can be related to several aspects. In this subsection we discuss model size, model complexity, and typical error patterns.

Model Size: The size of the model can be considered as a potential error determinant if the model is produced by a human modeler. Simon [31] points to the limited cognitive capabilities and concludes that humans act only rational to a limited extent. In the context of modeling, this argument would imply that human modelers lose track of all interrelations of a large model due to

Table 2

Branches of the SAP Reference Model. The columns $E_{av.}$, $F_{av.}$, $C_{av.}$, $A_{av.}$ refer to the mean number of events, functions, connectors, and arcs.

Branch	EPC	%	$E_{av.}$	$F_{av.}$	$C_{av.}$	$A_{av.}$	Cycle	Error	%
Asset Accounting	43	7.1%	13.9	4.0	5.2	23.3	0	7	16.3%
Benefits Administration	6	1.0%	9.5	3.3	5.8	19.7	3	0	0.0%
Compensation Management	18	3.0%	7.6	3.4	3.3	13.7	3	1	5.6%
Customer Service	41	6.8%	16.5	3.6	9.0	29.5	3	1	2.4%
Enterprise Controlling	22	3.6%	14.3	10.1	6.1	32.1	0	3	13.6%
Environment, Health, Safety	19	3.1%	3.5	2.7	1.2	7.0	0	0	0.0%
Financial Accounting	54	8.9%	13.0	4.0	5.1	21.8	0	3	5.6%
Position Management	0	0.0%	0.0	0.0	0.0	0.0	0	0	n.a.
Inventory Management	3	0.5%	15.0	7.0	6.0	28.0	2	0	0.0%
Organizational Management	5	0.8%	12.0	3.0	6.6	24.0	3	0	0.0%
Payroll	7	1.2%	5.7	3.1	2.1	11.4	0	1	14.3%
Personnel Administration	4	0.7%	7.3	1.5	4.0	12.3	0	0	0.0%
Personnel Development	10	1.7%	8.7	2.5	4.4	15.6	3	1	10.0%
Personnel Time Management	12	2.0%	10.8	3.0	5.3	19.5	1	2	16.7%
Plant Maintenance	35	5.8%	20.5	4.2	11.4	37.8	9	1	2.9%
Procurement	37	6.1%	6.7	3.5	2.7	12.4	0	2	5.4%
Product Data Management	26	4.3%	4.5	5.4	2.2	13.7	0	0	0.0%
Production	17	2.8%	8.8	3.0	2.9	13.7	0	1	5.9%
Production Planning	17	2.8%	5.7	2.9	3.0	11.5	0	0	0.0%
Project Management	36	6.0%	8.5	3.8	2.2	14.0	0	0	0.0%
Quality Management	20	3.3%	20.5	3.8	11.7	37.8	1	1	5.0%
Real Estate Management	6	1.0%	12.7	6.5	7.3	27.0	1	1	16.7%
Recruitment	9	1.5%	7.4	2.6	4.1	13.8	3	0	0.0%
Retail	1	0.2%	7.0	5.0	2.0	11.0	0	0	0.0%
Revenue & Cost Controlling	19	3.1%	16.5	10.2	7.9	36.0	1	1	5.3%
Sales & Distribution	76	12.6%	10.6	3.1	4.3	16.6	0	1	1.3%
Training & Event Management	12	2.0%	13.0	2.7	6.2	22.2	0	1	8.3%
Travel Management	1	0.2%	24.0	7.0	16.0	48.0	0	0	0.0%
Treasury	48	7.9%	10.5	3.5	4.5	18.1	0	6	12.5%
All 29 Branches	604	100%	11.5	4.0	5.2	20.8	33	34	5.6%

their limited cognitive capabilities, and then introduce errors that they would not insert in a small model. Accordingly, we define the following hypotheses:

- S_1 : A higher number of events E increases the error probability.
- S_2 : A higher number of functions F increases the error probability.
- S_3 : A higher number of connectors C increases the error probability.
- S_4 : A higher number of arcs A increases the error probability.

Model Complexity: Recent work by Cardoso [23] discusses complexity as an error source. Similar to large models, the modeler is expected to introduce errors more likely in complex models due to limited cognitive capabilities. Yet, complexity may differ from size, e.g., a large sequence may be less demanding for a modeler than small model containing several joins and splits. In EPCs complexity is introduced by *connectors*. This supports S_3. Moreover, two EPCs can have the same number of connectors, but differ in complexity if the second model introduces additional *arcs* between the connectors. Therefore, S_4 is also backed up from a complexity point of view. *Cycles* represent an additional aspect of complexity. Arbitrary cycles can lead to EPC models without clear semantics as shown in [24]. Cardoso introduces a *complexity metric* based on the observation that the three split connector types introduce a different degree of complexity. According to the number of potential post-states an AND-split is weighted with 1, an XOR-split with the number of successors n, and an OR-split with $2^n - 1$. We refer to the sum of all connector weights of an EPC as split-complexity SC (called Control-flow Complexity CFC in [23]). Analogously, we define the join-complexity JC as the sum of weighted join connectors based on the number of potential pre-states. Furthermore, we assume that a mismatch between potential post-states of splits and pre-states of joins can be modeled with the split-join-ratio $JSR = JC/SC$. Based on this we formulate the following hypotheses:

- C_1 : EPCs *with cycles* have a higher error probability than EPCs without.
- C_2 : A higher SC value of an EPC increases the error probability.
- C_3 : A higher JC value of an EPC increases the error probability.
- C_4 : A JSR value different from one increases the error probability.

Error Patterns: In contrast to hypotheses on complexity, error pattern point to structural properties of the model that may be the reason for problems. EPCs lack an explicit notion for the initial state, i.e. it is not clear in which combination of start events are allowed. This is reflected by the initial OR-split when translating an EPC to YAWL that covers all possible combinations. Clearly, this may be the source of misinterpretations by the modeler, and therefore the number of start events may influence the likelihood of errors being introduced. A similar observation may be made for the number of end events. A well-known source of errors are the so-called PT- and TP-handles in Petri nets [32]. A PT-handles starts with a place with multiple outgoing arcs joining later in a single transition. In terms of EPCs this means that an XOR-split connector corresponds to an AND-join connector. Clearly, this may indicate a deadlock problem: the process gets stuck just before AND-join. Similarly, an OR-split connector corresponding to an AND-join connector may be problematic. TP-handles are the reverse of PT-handles and start with a transition (AND-split) where outgoing arcs come together in a place (XOR-join). In terms of EPCs this corresponds to an AND-split or OR-split connector with a matching XOR-join connector. This establishes the following hypotheses:

- EP_1 : A higher number of start events increases the error probability.
- EP_2 : A higher number of end events increases the error probability.
- EP_3 : A higher number of XOR/OR-splits and AND-joins in an EPC increases the error probability.
- EP_4 : A higher number of AND/OR-splits and XOR-joins in an EPC increases the error probability.

Please note that EP_3 and EP_4 only indicate the possibility of a mismatch: if the numbers of splits and joins of the same type are high but equivalent, it could be that there is no mismatch. Still considering potential combinations of a high number of connectors implies several ways to introduce a mismatch. Table 3 summarizes the input variables that we will investigate. The table also shows how these variables can be linked to the discussed hypotheses.

3.3 Testing of Error Determinants

We now utilize the analysis table of the SAP reference model (cf. Fig. 1) to test the significance of our hypotheses. The potential determinants listed

Table 3

Potential Determinants for Errors in the SAP Reference Model

Symbol	Definition	Motivation
A	Number of Arcs	S_4
E_{start}	Number of Start Events	S_1, EP_1
E_{end}	Number of End Events	S_1, EP_2
E_{int}	Number of Internal Events	S_1
F	Number of Functions	S_1
AND_j	Number of AND joins	S_1, EP_3
AND_s	Number of AND splits	S_1, EP_4
XOR_j	Number of XOR joins	S_1, EP_4
XOR_s	Number of XOR splits	S_1, EP_3
OR_j	Number of OR joins	S_1
OR_s	Number of OR splits	S_1, EP_3, EP_4,
Cycle	if the EPC has cycles	C_1
SC	Split Complexity	C_2
JC	Join Complexity	C_3
JSR	Join-Split-Ratio	C_4

in Table 3 serve as input variables to explain the variance of the dependent variable “hasError”. As the dependent variable is binary, we use a logistic regression (logit) model. The idea of a logit model is to model the probability of a binary event by its odds, i.e., the ratio of event probability divided by non-event probability. These odds are defined as $logit(p_i) = ln(\frac{p_i}{1-p_i}) = \beta_0 + \beta_1 x_{1,i} + \cdots + \beta_k x_{k,i}$ for k input variables and i observations, i.e. EPC i in our context. From this follows that

$$p_i = \frac{e^{\beta_0+\beta_1 x_{1,i}+\cdots+\beta_k x_{k,i}}}{1+e^{\beta_0+\beta_1 x_{1,i}+\cdots+\beta_k x_{k,i}}}$$

The relationship between input and dependent variables is represented by an S-shaped curve of the logistic function that converges to 0 for $-\infty$ and to 1 for ∞ (see Figure 8). The cut value of 0.5 defines whether event or non-event is predicted. $Exp(\beta_k)$ gives the multiplicative change of the odds if the input variable β_k is increased by one unit, i.e. $Exp(\beta_k) > 1$ increases and $Exp(\beta_k) < 1$ decreases error probability.

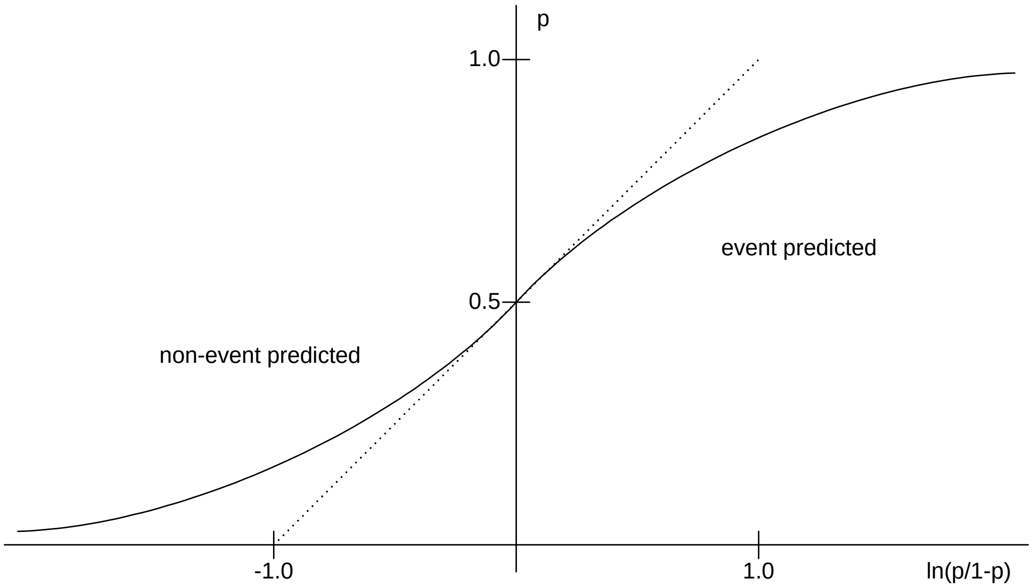

Fig. 8. S-shaped curve of the logistic regression model

The significance of the overall model is assessed by the help of two statistics. Firstly, the *Hosmer & Lemeshow* Test should be greater than 5% to indicate a good fit based on the difference between observed and predicted frequencies (cf. [33]). Secondly, *Nagelkerke's* R^2 ranging from 0 to 1 serves as a coefficient of determination indicating which fraction of the variability is explained [34]. Furthermore, each estimated coefficient of the logit model is tested using the *Wald* statistic, for being significantly different from zero. The significance should be less than 5%. We calculate the logistic regression model based on a stepwise introduction of those variables that provide the greatest increase in likelihood. For more details on logistic regression, see [33].

Our analysis was done in two steps. In the first step we analyzed the individual variables (univariate analysis) while in the second step we looked a combinations of variables (multivariate analysis).

As a first step we calculated univariate logit models for each of the 15 input variables. Each model for the 11 variables that indicate the number to elements of a specific type in the EPC had a Wald statistic at a significance level of 0.6% or better. The binary variable for cycles showed a significance of 10.6% in the Wald test which is not as good as the frequently used 5% significance level. The three complexity metrics all had a very poor Wald value with a significance between 70.8% to 78.1%. Accordingly, the null hypothesis that they have no impact on the odds of an error cannot be rejected. So based on the univariate logit models we can conclude that the various metrics related to the size of the model seem to be the best predictors for errors.

In a second step we tested multivariate logit models combining all input variables. Table 4 summarizes the results of this analysis. We started with all 15 variables yielding the results given in the "Complete Model" column. Together they are able to predict 95.2% correctly, i.e., without looking at the model and just observing the input variables, we can accurately predict whether a model has errors or not in 95.2 percent of the cases. Table 4 also shows the number of correctly predicted errors and the number of incorrectly predicted errors, e.g., using the "Complete Model" 3 of the 604 models were predicted to have errors but did not have any. Table 4 shows that in the "Complete Model" the number of OR-joins is significant (Wald sig. is 0.3%) and has a considerable impact (Exp(B) is 2.209). As SC and JC were both estimated to be 1 (having no impact on the odds), we reduced the model to 13 variables. The result is given in column "Without SC and JC". The other two columns list the model with the maximum number of variables that all have Wald sig. better than 11% ("8-Step Model") and better than 5% ("5-Step Model"), respectively. The columns show that the estimated coefficients have a stable tendency and a relatively stable value. All Hosmer&Lemeshow and Nagelkerke R^2 values

Table 4

Multivariate Logit Models based on potential Error Determinants

	Complete Model		Without SC and JC		8-Step Model		5-Step Model	
Coefficient	Exp(B)	Wald Sig.	Exp(B)	Wald Sig.	Exp(B)	Wald Sig.	Exp(B)	Wald Sig.
Constant	0.023	0.0%	0.028	0.0%	0.024	0.0%	0.025	0.0%
A	1.097	39.0%	1.081	47.8%	-	-	-	-
E_{start}	0.641	0.2%	0.666	0.4%	0.719	0.2%	0.844	2.4%
E_{end}	1.151	24.3%	1.057	63.2%	1.128	6.1%	-	-
E_{int}	1.069	70.6%	1.045	80.8%	1.151	0.5%	1.162	0.3%
F	0.906	36.8%	0.903	35.8%	-	-	-	-
AND_j	1.065	81.8%	1.190	51.6%	1.321	10.9%	-	-
AND_s	0.786	35.7%	0.932	77.8%	-	-	-	-
XOR_j	1.705	3.8%	1.795	2.3%	2.010	0.0%	1.559	0.9%
XOR_s	0.493	0.6%	0.589	2.4%	0.654	2.2%	-	-
OR_j	2.209	0.3%	2.067	0.5%	2.233	0.0%	1.939	0.1%
OR_s	0.432	0.6%	0.426	0.6%	0.473	0.2%	0.639	0.9%
Cycle	0.951	94.1%	0.990	98.8%	-	-	-	-
SC	1.000	59.3%	-	-	-	-	-	-
JC	1.000	97.2%	-	-	-	-	-	-
JSR	1.032	45.6%	1.023	60.3%	-	-	-	-
Hosmer&Lem. Sig.		10.3%		89.5%		62.9%		52.0%
Nagelkerke R^2		0.326		0.304		0.300		0.266
Correct Classif.		95.2%		95.2%		94.7%		95.0%
Correct Error Pred.		8		8		6		5
Wrong Error Pred.		3		3		4		1

indicate good fit of the statistical model to the data. The 8-Step model yields a prediction of 0.143 for our "Certificate Creation" EPC from the running example. This is below the 0.5 cut-off value and leads to an incorrect prediction of the model having no errors. The model with the highest prediction value (0.945) is a large EPC with 122 arcs, 24 connectors, 40 events, and 43 functions. This model includes errors which is correctly predicted.

The different multivariate logit models suggest the following conclusions. First, the *complexity metrics* proposed by [23] seem to have no impact on the odds of an error at all. The Wald test has both a bad significance and also predicts coefficients very close to zero. An explanation could be that OR-connectors get a weight that depends exponentially on the connector cardinality. Consider the example of an AND-split-join block with 5 parallel threads. Both SC and JC would result in a complexity metric of 1. Changing the connector types from AND to OR changes both metrics to 32. This great change in the metric based on state complexity obviously does not reflect the perceived conceptual complexity by the modeler. As the modeler is the one who introduces errors, these metrics seem to be misleading when used for the prediction of errors. Furthermore, the fact that a model includes *cycles* is not significant in the Wald statistic. Moreover, the number of *arcs* does not seem to have a huge impact on the odds, maybe because size is also captured by the number of other model elements and complexity by the number of connectors. The number of *start events* has a coefficient that reduces the odds. This might be related to the way how start events are used in the SAP reference models. There are several EPC models with lots of start events that are directly joined for representing alternative start triggers. This leads to a very simplistic join structure that is unlikely to produce errors. The coefficient for number of *functions* is not significantly different from zero with a tendency to a "negative" impact on the error probability. In contrast to that, both the number of *end and internal events* increase error probability, but not very strongly. Furthermore, it is interesting to see that all join *connectors* tend to have a "positive" impact on the odds of an error. The OR join has the highest coefficient of about 2. On the other hand, all split connectors have a "negative" impact. Interestingly, each pair of connectors has coefficients that have almost the same impact, but in a different direction. As an example, consider the coefficients for OR connectors of the 8-Step model. Introducing a pair of OR join and split connectors would have an impact on the odds of $0.473 * 2.233 = 1.056$. With respect to the error patterns of EP_3, introducing an XOR or OR split and an AND join increases error probability by $0.654 * 1.321 = 0.864$ or $0.473 * 1.321 = 0.625$, respectively. For EP_4 the values are above one if we consider the 13-variable model. Since not all coefficients are significant, an interpretation is difficult. Clearly speaking, there is no support for EP_3 and EP_4. Finally, the very small constant of about 0.025 indicates that the probability of an error is very small. This is consistent with the observation that you need at least a split and a join connector that do not match in order to introduce an error.

Beyond the significance of each individual coefficient, multivariate logistic regression appears to be a suitable tool to predict error probability in the SAP reference model. *Based on only 5 coefficients we are able to classify 95% of the EPCs correctly without looking into the model* (with a Nagelkerke R^2 of above 0.25). Accordingly, complexity seems to be a major source of error probability, yet not in shape of complexity metrics but rather related to the number of join connectors in the EPC.

4 Related Research

This section discusses the work that is most related for the research areas verification (Section 4.1) and quantitative analysis in process modeling (Section 4.2).

4.1 Verification

Since the mid-nineties, a lot of work has been done on the verification of process models, and in particular workflow models [35–39]. Sadiq and Orlowska [40] were among the first to point out that modeling a business process (or workflow) can lead to problems like livelock and deadlock. In their paper, they present a way to overcome syntactical errors, but they ignore the semantical errors. Nowadays, most work that is conducted is focusing on semantical issues, i.e., “will the process specified always terminate” and similar questions. The work on verification that has been conducted in the last decade can roughly be put into three categories: (1) verification of formal models, (2) verification of informal models, and (3) verification by design.

In the category *verification of formal models* we consider the work that has been done on the verification of modeling languages with formal semantics. One of the most prominent examples of such a language are Petri nets [26,27]. Especially in the field of *workflow management*, Petri nets have proven to be a solid theoretical foundation for the specification of processes. This, however, led to the need of verification techniques, tailored towards Petri nets that represent workflows. In the work of Van der Aalst and many others [29,41,42,12,43,30], these techniques are used extensively for verification of different classes of workflow definitions. Verification tools based on these approaches provide an answer in terms of "correct" or "incorrect". Besides Petri nets also other established formal languages have been used, e.g., process algebras, temporal logics and Turing machines. Moreover, some authors proposed the use of dedicated (typically graph based) languages. Examples are the metagraphs in [44] and the logic-based approach in [45,46].

However, not all modeling languages have formal semantics, in particular, UML activity diagrams and EPCs. The *verification of such informal models* can benefit from Petri net analysis techniques by translation. For EPCs several translations to Petri nets have been proposed, e.g. [13,47,48]. In our approach we utilize a translation to YAWL as reported in [25]. The formalization of EPCs as a state-transition-system is extensively discussed in [24]. It is shown that interacting OR-joins can lead to EPCs that do not have formal semantics. These EPCs are called unclean. In [49] an approach is presented to efficiently calculate the state space of a clean EPC, thereby providing executable semantics for the EPC.

The last category *verification by design* is somewhat of an outsider. Instead of verifying a model given in a specific language, it is also possible to define a language in such a way that the result is always correct. An example of such a modeling language is IBM MQSeries Workflow [39]. This language uses a specific structure for modeling, which will always lead to a correct and executable specification. However, modeling processes using this language requires advanced technical skills and the resulting model is usually far from intuitive.

Besides the three categories, there are some verification approaches that are more or less a combination of others. Consider for example the approach presented in [50], where EPCs are verified using an interactive verification approach. However, instead of generating a subclass of EPCs for which the approach works, the process designer or process owner is actively involved in the verification process by using his knowledge about the process which is not made explicit in the model. The latter is the reason why this approach could not be used for the automatic verification of the entire SAP reference model since it depends upon the knowledge of the process owners. The approach we use in this article, i.e. the WofYAWL approach, is described in detail in [10]. Again, this approach is somewhat of an outsider. The approach takes a model with a formal semantics (i.e., a YAWL model) to check relaxed-soundness which is a minimum correctness criterion for YAWL models. Still, there might be models that are relaxed-sound, but not correct against the more strict soundness criterion. Nevertheless, it finds errors in the YAWL model that should be corrected. By translating EPCs to YAWL models, we could use this approach.

4.2 Quantitative Research on Process Modeling

In contrast to the rich set of work on formal aspects of process modeling, only little research has been dedicated to quantitative aspects. In [51] the understandability of join and split representation in EPCs is compared to Petri nets from a modeler perspective. According to this study, users seem to understand the EPC notation easier. A recent survey reported in [1] identifies the most popular conceptual modeling languages and tools in Australia. Furthermore, the authors identify a set of motivations why modeling is used in practice and summarize prior quantitative work on observed advantages and disadvantages of modeling. Beyond that, we are not aware of quantitative research that aims at identifying determinants for errors in process models. There has been some research on complexity metrics for process models motivated by the idea that complexity would increase probability of errors [23]. While the empirical validation of complexity metrics for predicting software errors has been investigated for a while (see e.g. [52,22]), there is no evidence up to now for business process models.

To summarize this overview of related work, we point out that this article uniquely combines error detection based on formal methods with quantitative analysis of potential error determinants. This way, we have been able to provide a lower bound of 5.6% for the percentage of errors in the SAP reference model and evidence that complexity indeed has a significant impact on error probability.

5 Contributions & Limitations

In this article, we presented an approach to automatically identify errors in the SAP reference model. This formal analysis builds on a mapping from EPCs to YAWL and the analysis tool WofYAWL. It is one of the few studies using formal methods for quantitative research. We provided an in-depth analysis of errors in the SAP reference model which yields a lower bound for the number of errors (5.6% of the 604 EPCs). As far as we know, this is the first systematic analysis of the EPCs in the SAP reference model. Our findings demonstrate the need for formal analysis of process models in practice.

Moreover, we used a multivariate logistic regression model to test whether certain model characteristics related to complexity can serve as error determinants. Beyond the significance of each individual coefficient we can conclude that multivariate logistic regression appears to be a suitable tool to predict error probability in the SAP reference model. Based on only 5 coefficients we were able to classify 95% of the EPCs correctly, i.e., *without* analyzing the model in detail we can predict the presence of an error quite accurately based on simple criteria. Therefore, complexity seems to be a major source of error probability, yet not in the shape of the complexity metrics defined in [23] but rather related to the number of joins in the EPC. This is an important finding that motivates further research on the measurement of business process model complexity.

Yet, our approach still has some limitations. It is a shortcoming for the estimation of a logit model that WofYAWL finds only those errors that can be related to relaxed soundness, and not those that affect the more strict soundness criterion. Therefore, we need further research on automatic identification of errors. Beyond that, we need to analyze errors in business processes that have been modeled in different languages than EPCs. While the relaxed soundness analysis could be also applied to languages like UML activity diagrams and BPMN models, the different set of modeling elements might have an impact on the contribution of different elements to error probability. Future research will also have to investigate how those potential determinants that are not significant in the test perform in the context of other business process model samples. Accordingly, we aim to reuse this research design for other large enterprise models in order to test whether the coefficients are stable. A systematic analysis of more large enterprise models could result in a theory explaining when human modelers are likely to introduce errors in a process model. Such a theory would offer valuable insights for the teaching of process modeling languages in companies and universities making people aware of situations where errors occur more frequently. The 5.6% found in this paper can be considered as a first benchmark for error probability in business process model collections.

References

[1] I. Davies, P. Green, M. Rosemann, M. Indulska, S. Gallo, How do practitioners use conceptual modeling in practice?, Data & Knowledge Engineering 58 (3) (2006) 358–380.

[2] T. Curran, G. Keller, A. Ladd, SAP R/3 Business Blueprint: Understanding the Business Process Reference Model, Enterprise Resource Planning Series, Prentice Hall PTR, Upper Saddle River, 1997.

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

[4] P. Fettke, P. Loos, Classification of reference models - a methodology and its application, Information Systems and e-Business Management 1 (1) (2003) 35–53.

[5] K. Lang, M. Schmidt, Workflow-supported organizational memory systems: An industrial application., in: 35th Hawaii International International Conference on Systems Science (HICSS-35 2002), CD-ROM / Abstracts Proceedings, IEEE Computer Society, 2002, p. 208.

[6] J. Mendling, G. Neumann, M. Nüttgens, Yet Another Event-Driven Process Chain, in: Proceedings of BPM 2005, Vol. 3649 of Lecture Notes in Computer Science, 2005.

[7] M. Rosemann, W. van der Aalst, A Configurable Reference Modelling Language, Information Systems 32 (2007) 1–23.

[8] O. Thomas, A.-W. Scheer, Tool support for the collaborative design of reference models - a business engineering perspective., in: 39th Hawaii International International Conference on Systems Science (HICSS-39 2006), CD-ROM / Abstracts Proceedings, 4-7 January 2006, Kauai, HI, USA, IEEE Computer Society, 2006.

[9] G. Keller, M. Nüttgens, A. Scheer, Semantische Processmodellierung auf der Grundlage Ereignisgesteuerter Processketten (EPK), Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89 (in German), Saarbrücken (1992).

[10] H. Verbeek, W. van der Aalst, A. ter Hofstede, Verifying workflows with cancellation regions and or-joins: An approach based on relaxed soundness and invariants, The Computer Journal 50 (3) (2007) 294–314.

[11] W. van der Aalst, A. ter Hofstede, YAWL: Yet Another Workflow Language, Information Systems 30 (4) (2005) 245–275.

[12] J. Dehnert, P. Rittgen, Relaxed Soundness of Business Processes, in: K. Dittrich, A. Geppert, M. Norrie (Eds.), Proceedings of CAiSE 2001, Vol. 2068 of LNCS, Springer-Verlag, Berlin, 2001, pp. 157–170.

[13] J. Dehnert, W. van der Aalst, Bridging The Gap Between Business Models And Workflow Specifications, International J. Cooperative Inf. Syst. 13 (3) (2004) 289–332.

[14] J. Dehnert, A. Zimmermann, On the suitability of correctness criteria for business process models., in: W. van der Aalst, B. Benatallah, F. Casati, F. Curbera (Eds.), Business Process Management, 3rd International Conference, BPM 2005, Nancy, France, September 5-8, 2005, Proceedings, Vol. 3649 of Lecture Notes in Computer Science, 2005, pp. 386–391.

[15] A. Basu, A. Kumar, Research commentary: Workflow management issues in e-business, Information Systems Research 13 (1) (2002) 1–14.

[16] B. Boehm, Software Engineering Economics, Prentice-Hall, Englewood Cliffs, 1981.

[17] Y. Wand, R. Weber, Research Commentary: Information Systems and Conceptual Modeling - A Research Agenda, Information Systems Research 13 (4) (2002) 363–376.

[18] T. McCabe, A complexity measure, IEEE Transactions on Software Engineering 2 (4) (1976) 308–320.

[19] T. McCabe, C. Butler, Design complexity measurement and testing, Communications of the ACM 32 (1989) 1415–1425.

[20] M. H. Halstead, Elements of Software Science., Vol. 7 of Operating, and Programming Systems Series, Elsevier, Amsterdam, 1977.

[21] S. Henry, D. Kafura, Software structure metrics based on information-flow, IEEE Transactions On Software Engineering 7 (5) (1981) 510–518.

[22] N. E. Fenton, N. Ohlsson, Quantitative analysis of faults and failures in a complex software system, IEEE Transactions on Software Engineering 26 (8) (2000) 797–814.

[23] J. Cardoso, Control-flow Complexity Measurement of Processes and Weyuker's Properties, in: 6th International Enformatika Conference, Transactions on Enformatika, Systems Sciences and Engineering, Vol. 8, 2005, pp. 213–218.

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

[25] J. Mendling, M. Moser, G. Neumann, Transformation of yEPC Business Process Models to YAWL, in: Proceedings of the 21st Annual ACM Symposium on Applied Computing, Vol. 2, ACM, Dijon, France, 2006, pp. 1262–1267.

[26] J. Desel, J. Esparza, Free Choice Petri Nets, Vol. 40 of Cambridge Tracts in Theoretical Computer Science, Cambridge Univ. Press, Cambridge, UK, 1995.

[27] T. Murata, Petri Nets: Properties, Analysis and Applications, Proceedings of the IEEE 77 (4) (1989) 541–580.

[28] W. Reisig, G. Rozenberg (Eds.), Lectures on Petri Nets I: Basic Models, Vol. 1491 of LNCS, Springer-Verlag, Berlin, 1998.

[29] W. van der Aalst, Workflow Verification: Finding Control-Flow Errors using Petri-net-based Techniques, in: W. Aalst, J. Desel, A. Oberweis (Eds.), Business Process Management: Models, Techniques, and Empirical Studies, Vol. 1806 of LNCS, Springer-Verlag, Berlin, 2000, pp. 161–183.

[30] H. Verbeek, T. Basten, W. van der Aalst, Diagnosing Workflow Processes using Woflan, The Computer Journal 44 (4) (2001) 246–279.

[31] H. Simon, Sciences of the Artificial, 3rd Edition, The MIT Press, 1996.

[32] J. Esparza, M. Silva, Circuits, handles, bridges and nets, in: G. Rozenberg (Ed.), Advances in Petri Nets 1990, Vol. 483, 1990, pp. 210–242.

[33] D. Hosmer, S. Lemeshow, Applied Logistic Regression, 2nd Edition, John Wiley & Sons, 2000.

[34] N. Nagelkerke, A note on a general definition of the coefficient of determination, Biometrika 78 (3) (1991) 691–692.

[35] W. van der Aalst, K. van Hee, Workflow Management: Models, Methods, and Systems, MIT press, Cambridge, MA, 2002.

[36] M. Dumas, W. van der Aalst, A. ter Hofstede, Process-Aware Information Systems: Bridging People and Software through Process Technology, Wiley & Sons, 2005.

[37] D. Georgakopoulos, M. Hornick, A. Sheth, An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure, Distributed and Parallel Databases 3 (1995) 119–153.

[38] A. Kumar, J. Zhao, Workflow Support for Electronic Commerce Applications, Decision Support Systems 32 (3) (2002) 265–278.

[39] F. Leymann, D. Roller, Production Workflow: Concepts and Techniques, Prentice-Hall PTR, Upper Saddle River, New Jersey, USA, 1999.

[40] W. Sadiq, M. Orlowska, Applying graph reduction techniques for identifying structural conflicts in process models., in: M. Jarke, A. Oberweis (Eds.), Advanced Information Systems Engineering, 11th International Conference CAiSE'99, Heidelberg, Germany, June 14-18, 1999, Proceedings, Vol. 1626 of Lecture Notes in Computer Science, Springer, 1999, pp. 195–209.

[41] W. van der Aalst, A. ter Hofstede, Verification of Workflow Task Structures: A Petri-net-based Approach, Information Systems 25 (1) (2000) 43–69.

[42] W. van der Aalst, A. Kumar, XML Based Schema Definition for Support of Inter-organizational Workflow, Information Systems Research 14 (1) (2003) 23–46.

[43] K. van Hee, N. Sidorova, M. Voorhoeve, Soundness and Separability of Workflow Nets in the Stepwise Refinement Approach, in: W. Aalst, E. Best (Eds.), Application and Theory of Petri Nets 2003, Vol. 2679 of LNCS, Springer-Verlag, Berlin, 2003, pp. 335–354.

[44] A. Basu, R. Blanning, A Formal Approach to Workflow Analysis, Information Systems Research 11 (1) (2000) 17–36.

[45] H. Bi, J. Zhao, Applying Propositional Logic to Workflow Verification, Information Technology and Management 5 (3-4) (2004) 293–318.

[46] H. Bi, J. Zhao, Process Logic for Verifying the Correctness of Business Process Models, in: Proceedings of the International Conference on Information Systems (ICIS 2004), Association for Information Systems, 2004, pp. 91–100.

[47] W. van der Aalst, Formalization and Verification of Event-driven Process Chains, Information and Software Technology 41 (10) (1999) 639–650.

[48] P. Langner, C. Schneider, J. Wehler, Petri Net Based Certification of Event driven Process Chains, in: J. Desel, M. Silva (Eds.), Application and Theory of Petri Nets 1998, Vol. 1420 of LNCS, Springer-Verlag, Berlin, 1998, pp. 286–305.

[49] N. Cuntz, J. Freiheit, E. Kindler, On the Semantics of EPCs: Faster Calculation for EPCs with Small State Spaces, in: M. Nüttgens, F. Rump (Eds.), Proceedings of Fourth Workshop on Event-Driven Process Chains (EPK 2005), Gesellschaft für Informatik, Bonn, Hamburg, Germany, 2005, pp. 7–23.

[50] B. van Dongen, H. Verbeek, W. van der Aalst, Verification of EPCs: Using reduction rules and Petri nets, in: Conference on Advanced Information Systems Engineering (CAiSE 2005), Vol. 3520, 2005, pp. 372–386.

[51] K. Sarshar, P. Loos, Comparing the control-flow of epc and petri net from the end-user perspective., in: W. Aalst, B. Benatallah, F. Casati, F. Curbera (Eds.), Business Process Management, 3rd International Conference, BPM 2005, Nancy, France, September 5-8, 2005, Proceedings, LNCS 3649, 2005, pp. 434–439.

[52] V. Basili, B. Perricone, Software errors and complexity: an empirical investigation, Communications of the ACM 27 (1) (1984) 42–52.