Who Is Behind the Model? Classifying Modelers Based on Pragmatic Model Features

Andrea Burattin[1(✉)], Pnina Soffer[2], Dirk Fahland[3], Jan Mendling[4], Hajo A. Reijers[3,5], Irene Vanderfeesten[3], Matthias Weidlich[6], and Barbara Weber[1,7]

[1] Technical University of Denmark, Kgs. Lyngby, Denmark
andbur@dtu.dk
[2] University of Haifa, Haifa, Israel
[3] Eindhoven University of Technology, Eindhoven, The Netherlands
[4] Vienna University of Economics and Business, Vienna, Austria
[5] Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
[6] Humboldt-University, Berlin, Germany
[7] University of Innsbruck, Innsbruck, Austria

Abstract. Process modeling tools typically aid end users in generic, non-personalized ways. However, it is well conceivable that different types of end users may profit from different types of modeling support. In this paper, we propose an approach based on machine learning that is able to classify modelers regarding their expertise while they are creating a process model. To do so, it takes into account pragmatic features of the model under development. The proposed approach is fully automatic, unobtrusive, tool independent, and based on objective measures. An evaluation based on two data sets resulted in a prediction performance of around 90%. Our results further show that all features can be efficiently calculated, which makes the approach applicable to online settings like adaptive modeling environments. In this way, this work contributes to improving the performance of process modelers.

Keywords: Process modeling · Classification of modelers
Model layout

1 Introduction

Process models play an important role in the analysis, redesign, and implementation of business processes [1,2]. The creation of process models is a design activity [3], in which a modeler constructs a mental model of a given domain and externalizes it using a specific modeling tool (including the modeling notation) [4]. This design activity involves deciding which elements to use, which names to give them, where to position them, and how to connect them. We also refer to this activity as modeling.

M. Weske et al. (Eds.): BPM 2018, LNCS 11080, pp. 322–338, 2018.
https://doi.org/10.1007/978-3-319-98648-7_19

Modeling is not for free. The activity of creating a model imposes a substantial cognitive load on the limited information processing capacity of the modeler's brain [5]. In particular, cognitive load depends upon various factors including task characteristics, modeler characteristics, and tool characteristics. Modeling research has hardly considered the latter so far [6], and indeed, tools do not anticipate personal differences when they support the modeler [7–10]. However, personalized support could be highly beneficial for novices who require tips and guidance, while experts would perceive this as a distraction. If the profile of the modeler is known, such support can significantly improve performance [11].

In this paper, we lay foundations towards a personalization of modeling tool support. Our key idea is to support an on-the-fly classification of modelers by their expertise level while they interact with the tool. To this end, we identify a set of pragmatic modeling features that presumably reflect the expertise of the modeler in activity-centric, flow-based process models with AND/XOR gateways. We evaluate the relevance of these features using real-world modeling traces of BPMN models in order to classify the modelers as novices or experts. With a classification accuracy of 90%, our results demonstrate the feasibility of personalized support.

The remainder of the paper is structured as follows: Sect. 2 presents background information and related work; Sect. 3 describes our approach to classify modelers. Section 4 evaluates the classification technique on two real datasets and Sect. 5 concludes the paper.

2 Background and Related Work

2.1 Process Modeling as a Design Activity

Creating a process model constitutes a complex cognitive design activity. During this design activity a process modeler solves a problem of how to represent a described process as a process model, using the syntax of a specific modeling language. As a problem solving task, this entails the formation of a mental representation of the problem domain and externalizing this representation as a process model [4,12]. Doing this, the modeler interacts with the modeling environment to create the process model. More precisely, the modeler performs a sequence of modeling interactions (like the creation of an activity or an edge or the movement of an element) resulting into (intermediate) models [13]. The resulting (intermediate) models can be characterized by properties referring to their syntax, semantics, or pragmatics [14]. A graphical representation of these interactions over time, and possible artifacts obtained, is given in Fig. 1.

2.2 Expertise in Modeling

Differences between novices and experts have been intensively studied in the context of various tasks and artifacts. However, throughout the body of research that deals with expertise-related differences, no single and agreed upon criterion

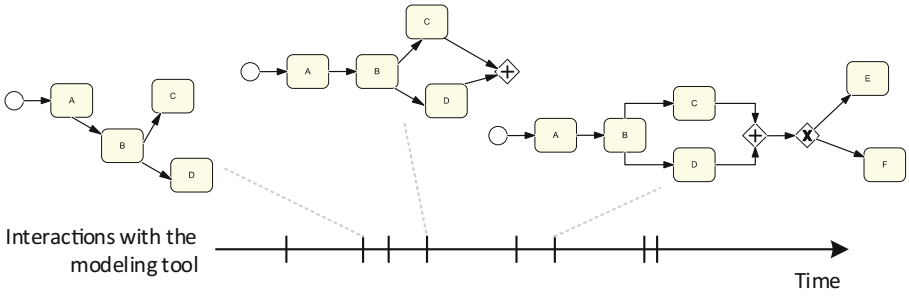


Fig. 1. Interactions with a modeling tool result in different intermediate models.

is used for distinguishing experts from novices. In fact, in [15], the authors define and test the effect of different dimensions of expertise, such as familiarity with a modeling language, intensity of modeling engagement, and knowledge of modeling concepts. In particular, they examined the distinction between students and practitioners, indicating that students exhibit different patterns of interaction with models as compared to practitioners. Bearing all this in mind, we rely on studies that concern differences in task performance between experts and novices for establishing expected differences in our study.

Concerning problem solving in Physics, in [16] authors discuss differences in the strategies employed by experts and novices, related to the differences in the mental problem representation they form and the retrieval of appropriate concepts and solution procedures from long term memory. These differences between novices and experts, in the availability and ease of retrieval of relevant concepts and solution strategies, pertain to many other areas.

In the area of conceptual modeling and process modeling, two main tasks are distinguished: reading (understanding) a model and creating a model. The differences between experts and novices in reading and understanding models have been studied [17], with a general indication that novices and experts have different notational needs [18]. Novices have more difficulties to recognize semantic patterns from graphics and tie them to long term memory concepts. Thus, in general, reading a model entails a higher cognitive load for a novice than for an expert [6,15]. In particular, this cognitive load and the understanding performance can be affected by graphical properties and layout of the model [15].

Novice and expert difference in creating models have also been indicated. For conceptual models, in [19], authors found that experts focus on generating a holistic understanding of the problem, making abstractions of problem characteristics by categorizing problem descriptions before developing the solution. Novices had difficulties in integrating parts of the problem descriptions and mapping them into knowledge structures. In [20], authors present the results of an empirical study aimed at identifying the most typical set of errors frequently committed by novice systems analysts in four commonly used UML artifacts.

In general, these errors can be interpreted as indicating a difficulty in making abstractions, which leads to a focus on specific functional details rather than on solution principles. A good use of graphical cues and layout creation by experts is indicated by [18] in the context of graphical programming. They claim that expert's categorization skills and ability to organize information on the basis of underlying abstractions are reflected in the expert's ability take advantage of secondary notation cues to enable them to recognize sub-term groupings.

Generally speaking, the above discussed differences between experts and novices can lead to two main conclusions. First, experts and novices would benefit from different kinds of support and guidance while creating a model. Second, it should be possible to distinguish and identify whether a certain model is being created by a novice or by an expert, as elaborated next.

3 Identifying the Expertise Level of Process Modelers

Many research efforts over the years have been devoted to personalizing systems based on user properties, organized in a *user model* (e.g., [21]). In this paper, as a first step towards a personalized modeling support, we aim at the most basic, simplest possible user model: a binary classification into *novice* or *expert*.

3.1 Classifying Modelers: Requirements and Design Considerations

Following user modeling literature [21], we identify 4 requirements for an approach for the classification of modelers expertise level:

- R1 The approach should be based on objective measures, rather than on modelers' self-assessment of their expertise level;
- R2 The approach should be unobtrusive towards the end user, and not involve additional efforts of modelers (e.g., for providing information);
- R3 The approach should work online, and be applicable to intermediate (incomplete) models, since modelers are likely to learn and improve their skills over time;
- R4 The approach should not depend on a particular modeling tool.

With these four requirements, we turn to assess available approaches for classifying a modeler in terms of expertise level.

One approach is to rely on self-assessment and to ask the modeler to classify himself/herself, e.g., by choosing a predefined profile in the modeling environment. Such an assessment is, however, neither based on objective measures (R1), nor applicable in an online setting since it is not automated (R3).

A second approach is to elicit this information based on a questionnaire regarding relevant modeler-specific features (like modeling experience, domain knowledge, cognitive abilities). For example, [11] used a questionnaire-based modeler's cognitive style for a personalized modeling support. The need to provide such information might, however, raise privacy issues, and seem obtrusive by

the modeler (R2). Moreover, due to the manual efforts needed it is not applicable as an online approach (R3).

A third approach is the classification of modelers based on neuro-physiological measures [22]. For example, [23] used the Alpha and Theta signals of an EEG to quantify programmer's expertise. While such an approach can be automated and is based on objective measures of cognitive load, it is intrusive and is not applicable outside of a lab setting (R2).

A fourth approach is classification based on differences in modeling behavior derived from the recorded interactions with the modeling platform. For example, [24] showed that the presence of prior domain knowledge facilitates the creation of an internal representation of the process to be modeled and is associated with shorter initial comprehension phases. Moreover, Martini et al. [25] showed that inexperienced modelers had significantly more comprehension and modeling phases when compared to more experienced modelers. The drawback of relying on detected modeling behavior is, however, its tool dependence (R4).

Finally, a fifth approach is using the (intermediate) modeling artifact as a basis for classification. Considering only model features provides tool independence, since the properties that can serve as features are derived from the model without a need for knowledge about tool interactions. Features related to syntax and semantics (e.g., presence of deadlocks and lack of synchronization) are less suited for online settings. This is since existing metrics assume certain properties of the model (e.g., all elements are fully connected and the model is sound), assumptions that are mainly applicable to complete models rather than to intermediate ones (R3). In contrast, pragmatic properties, that capture the alignment of process model elements or the way gateways are used, fulfill all the above mentioned requirements and will be used as a basis for our classification approach.

3.2 Overview of Our Approach

Classification problems have been extensively studied in the literature [26] and most approaches assume a *feature vector* as input (cf. Sect. 3.3 for the considered features). Figure 2 depicts the general idea proposed by our approach: process modelers interact with the modeling tool to construct a BPMN diagram. Examples of interactions are the creation of an activity or edge or the movement of an element. Hereby, the process model gradually evolves over time resulting into different (intermediate) models. Input to the classification is one such (intermediate) model. In step ① the set of features used for the classification is extracted from the model. Then, in step ②, this features vector is given as input to a trained classification model, which returns the likelihood that the model (i.e., the features vector) has been created by a novice or by an expert.

3.3 Feature Engineering

Feature engineering is considered the "art of creating predictor variables" [27]: this human-driven process requires iterations of brainstorming possible features and studying their impact on the quality of the model. We manually inspected

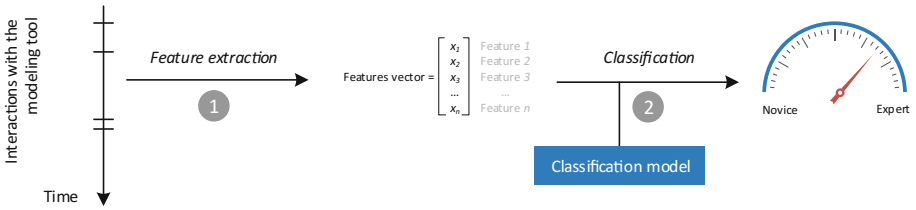


Fig. 2. Our approach to classify the expertise level of process modelers.

several models generated by both experts and novices and we investigated the most relevant differences. Moreover, as a consequence of requirement of being applicable in an online setting (R3) (i.e., support for online measurements) features should be computed efficiently. Thus, we also took into consideration the complexity of computing them.

The first group of features we identified considers the alignment of fragments. The reasoning behind these features is that the alignment of fragments helps model comprehension [28]. Poorly aligned fragments obfuscate the process model and make it difficult to recognize the patterns used. In the context of this paper, we define a *fragment* as a SESE component [29] with at least two tasks. We say that a fragment is *aligned* if the coordinates of its entry and its exit components (i.e., the coordinates of the center of entry and exit components) are within a certain threshold. Formally, two elements a and b located at (x_a, y_a) and (x_b, y_b) are aligned if $\min\{|x_a - x_b|, |y_a - y_b|\} \leq 20$px. Based on these definitions we can define the following features:

- F1 *Alignment of fragments*: ratio of aligned SESE fragments over the total number of fragments in the BPMN model;
- F2 *Percentage of activities in aligned fragments*: ratio of activities belonging to aligned SESE fragments over the total number of activities;
- F3 *Percentage of activities in not aligned fragments*: ratio of activities belonging to not aligned SESE fragments over the total number of activities.

The next group of features considers the type and usage of the gateways. The usage of implicit gateways as well as the reuse of gateways has been pointed out as bad modeling practice [28,30,31]. Specifically, we call a gateway *explicit* if it is represented as a BPMN gateway element. An *implicit* gateway, in turn, is a BPMN task element with more than 1 entering (or exiting) connections (i.e., sequence flows). Finally, a *reused* gateway is a gateway (either implicit or explicit) which serves as both split and join, i.e., it has more than 1 incoming and more than 1 outgoing edges. Based on these definitions we implemented the following features:

- F4 *Number of explicit gateways*: the total number of explicit gateways in the BPMN model;
- F5 *Number of implicit gateways*: the total number of implicit gateways in the BPMN model;

F6 *Number of reused gateways*: the total number of reused gateways (either implicit or explicit) in the BPMN model.

Moreover, we have a feature group concerning the style of the edges in a model, which is perceived as a relevant visual feature [31,32]. We distinguish between *edges* and *segments*: an edge consists of at least one segment and each bendpoint introduces one additional segment into the corresponding edge. Then we can define the following features:

F7 *Percentage of orthogonal segments*: ratio of the segments that are "orthogonal" (i.e., either vertical or horizontal within a threshold of 10px) over the total number of segments;
F8 *Percentage of crossing edges*: ratio of edges that are crossed by some other edge over the total number of edges.

The last set of features considers the process "as a whole" and therefore concerns more global properties:

F9 *M-BP*: this measure (in the interval $[0, 1]$) computes the extent to which the layout of the model is consistent with the temporal logical ordering of corresponding activities [31,33], computed using the algorithm in [34];
F10 *Number of ending points*: the number of nodes with at least one incoming edge, but no outgoing connection [28].

In sum, a *modeling session*, i.e., the modeling exercise of one person, is a series of model snapshots $m_1, m_2, m_3, \ldots$ over time, also called *intermediate models*. Each model, in turn, is characterized by a vector of 10 numerical features (we say that $m_1, m_2, m_3, \ldots$ is *represented* by $\boldsymbol{F}_1, \boldsymbol{F}_2, \boldsymbol{F}_3, \ldots$), extracted in step ①. Those will be used to distinguish the expertise level of modelers.

3.4 Model Classification

For a given intermediate model, the described features were calculated and passed in step ② to a classification model. We used neural networks [35] as classification mechanism. This well studied classification model can be used to approximate any discrete-valued target functions: the "universal approximation theorem" [36] proves that multilayer feedforward networks with as few as a single hidden layer are universal approximators and therefore represent the most general tool available[1]. What is interesting of these models is their capability of dealing with complex decomposition of high dimensional spaces into smaller ones (in our case, we move from a 10-dimensions space to a binary problem). Additionally, by introducing specific topologies (i.e., hidden layers) it is possible to increase the chances of making the problem easier to solve (i.e., transforming the data space into a linear separable one).

We used a feed-forward neural network, with one hidden layer comprising 50 neurons. The input layer contains 10 neurons, one for each feature of the vectorial

[1] See external appendix for details: https://doi.org/10.5281/zenodo.1251633.

representation of the model. The output layer contains 2 neurons, whose values distinguish between the two classes (i.e., *novice* or *expert*). The rationale behind the topology of the network comes from an experimental phase, where different configurations have been verified: there is no scientific result defining a generally optimal structure. Still, there are results suggesting that 1 hidden layer can be enough [37]. Concerning the number of units, literature [38] suggests a number of hidden nodes proportional to the number of training samples over a multiple (between 2–10) of the sum of input/output nodes. In our case, we decided to focus on our smallest dataset, with 1000 samples, and 2 as multiplier.

4 Evaluation

In this section, we evaluate the accuracy of the classification for predicting if the modeler is a novice or an expert. As evaluation input, we use data sets that document modeling processes of students (as proxies for novices) and practitioners (as proxies for experts). Note that our approach is independent of the question whether students and practitioners always represent valid proxies. For a recent discussion of student/practitioner differences, refer to [15].

The described approach has been implemented in a Java application[2] and tested on 2 real datasets[3]. For the implementation, we used the libraries of the Weka toolkit[4] with the corresponding multilayer perceptron. The multilayer perceptron was trained with a learning rate of 0.3 and a learning momentum of 0.2. Additionally, we set the number of training epochs to 500.

4.1 Description of Datasets

We performed our tests on datasets collected in several modeling sessions in 2010. Novice data was collected through the participation of students from Eindhoven University of Technology. Expert data, in turn, was collected as part of a Dutch BPM round-table event in Eindhoven, as well as in Berlin with practitioners[5]. In both settings, the experts were recruited from our network of industry practitioners, who where experienced modelers and highly familiar with BPMN. The modeling sessions were ran at the universities in a controlled setting. Participants were aware that the exercises were meant to assess their competence, but did not know that the tool tracked each modeling step. A textual description of the processes to be modeled was provided and subjects were asked to model a BPMN model representing the described process. In this paper we analyze data referring to two process descriptions.

[2] The complete source code of the implementation is available at https://github.com/DTU-SPE/ExpertisePredictor4BPMN.

[3] The dataset is available at https://doi.org/10.5281/zenodo.1194780.

[4] See http://www.cs.waikato.ac.nz/ml/weka/.

[5] Please note that the data collected from the practitioners has not been published before. Moreover, the model features used as basis for this paper have not been reported before, neither for students nor for practitioners.

Table 1. Number of modeling sessions and (intermediate) models for each experiment and expertise level included in our datasets.

	Experts		Novices	
	Sessions	Interm. models	Sessions	Interm. models
mortgage-1	31	7299	144	36141
pre-flight	39	4856	118	14147

In the first model (called pre-flight) participants were asked to represent the steps conducted by an airplane's crew before take-off. The reference implementation contains 12 activities and 10 gateways[6]. In the second model (called mortgage-1) participants were asked to represent a mortgage application process with 26 activities and 20 gateways (see footnote 6). While the pre-flight process is fairly simple and only comprises sequences, parallel branches and several optional activities without any nesting, the mortgage-1 example is more complex. It contains a long loop back as well as several levels of nesting depth. In addition, the mortgage-1 process has several outcomes, i.e., rejection because of pre-existing mortgage, rejection through employee, offer not updated, and customer accepts offer. For modeling these processes we used Cheetah [39] which is able to record all interactions with the modeling platform and allows to reconstruct all intermediate models. Thus, for each modeling session (i.e., a single modeling exercise) we have multiple (intermediate) models based on which the features were calculated.

The number of modeling sessions and the number of available models are reported in Table 1. As we can see, the actual number of sessions and models differs between expertise levels and between modeling tasks. For this reason, we used a small fragment of randomly selected models for each experiment.

Before turning to the results of the classification, we compare the differences between the 2 groups of users (i.e., novices and experts) in our 2 datasets (considering intermediate models created during the last 70% of the modeling session). Specifically, we used the Mann-Whitney U Test to understand whether the features described in Sect. 3.3 are proper discriminators of the 2 groups. Results of the test are reported in Table 2 and clearly show that statistically significant differences between novices and experts exist for all 10 considered features. Table 3 reports the descriptive statistics for the two analyzed datasets with mean, standard deviation (SD) and standard error (SE) for each feature and both experts and novices. The descriptive statistics clearly depict, for both datasets (i.e., mortgage-1 and pre-flight), that experts prefer to align elements (cf. values of F1, F2, F3), experts prefer explicit gateways over implicit ones (cf. F4 and F5) and they reuse less gateways when compared to novices (cf. F6). Additionally, novices model less orthogonal segments (cf. F7) and also keep the layout less consistent with the temporal logical ordering of activities (cf. F9).

[6] Graphical representations on the appendix: https://doi.org/10.5281/zenodo.1251633.

Table 2. Mann-Whitney U Test on mortgage-1 and pre-flight datasets. The feature codes refer to the descriptions in Sect. 3.3.

(a) mortgage-1

Feature	W	p
F1	1.524e+8	< .001
F2	1.459e+8	< .001
F3	1.199e+8	< .001
F4	1.572e+8	< .001
F5	1.179e+8	< .001
F6	1.359e+8	< .001
F7	1.645e+8	< .001
F8	1.034e+8	< .001
F9	1.688e+8	< .001
F10	1.627e+8	< .001

(b) pre-flight

Feature	W	p
F1	3.650e+7	< .001
F2	3.934e+7	< .001
F3	3.339e+7	< .001
F4	4.128e+7	< .001
F5	3.252e+7	< .001
F6	3.257e+7	< .001
F7	3.850e+7	< .001
F8	3.463e+7	< .001
F9	3.870e+7	< .001
F10	3.331e+7	< .001

Table 3. Descriptive group statistics for experts and novices for the two datasets analyzed. The feature codes refer to the descriptions in Sect. 3.3.

Feature	Group	mortgage-1			pre-flight		
		Mean	SD	SE	Mean	SD	SE
F1	Experts	0.862	0.256	0.003	0.817	0.311	0.004
	Novices	0.807	0.261	0.001	0.764	0.361	0.003
F2	Experts	0.459	0.17	0.002	0.505	0.243	0.003
	Novices	0.434	0.177	0.0009	0.441	0.259	0.002
F3	Experts	0.09	0.164	0.002	0.078	0.151	0.002
	Novices	0.1	0.151	0.0008	0.098	0.188	0.002
F4	Experts	11.901	4.54	0.053	6.84	2.665	0.038
	Novices	10.194	4.468	0.024	5.937	2.515	0.021
F5	Experts	1.309	1.487	0.017	0.371	0.845	0.012
	Novices	1.576	1.614	0.008	0.495	1.079	0.009
F6	Experts	0.344	0.615	0.007	0.501	1.079	0.015
	Novices	0.316	0.627	0.003	0.471	0.773	0.006
F7	Experts	0.712	0.223	0.003	0.572	0.267	0.004
	Novices	0.603	0.179	0.0009	0.494	0.18	0.002
F8	Experts	0.008	0.026	0.0003	0.012	0.041	0.0006
	Novices	0.022	0.044	0.0002	0.008	0.035	0.0003
F9	Experts	0.95	0.067	0.0008	0.95	0.103	0.001
	Novices	0.877	0.126	0.0007	0.906	0.125	0.001
F10	Experts	2.743	1.14	0.013	1.598	0.88	0.013
	Novices	2.267	1.012	0.005	1.64	0.911	0.008

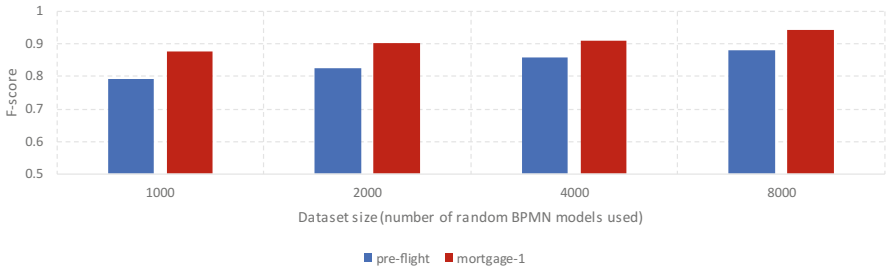
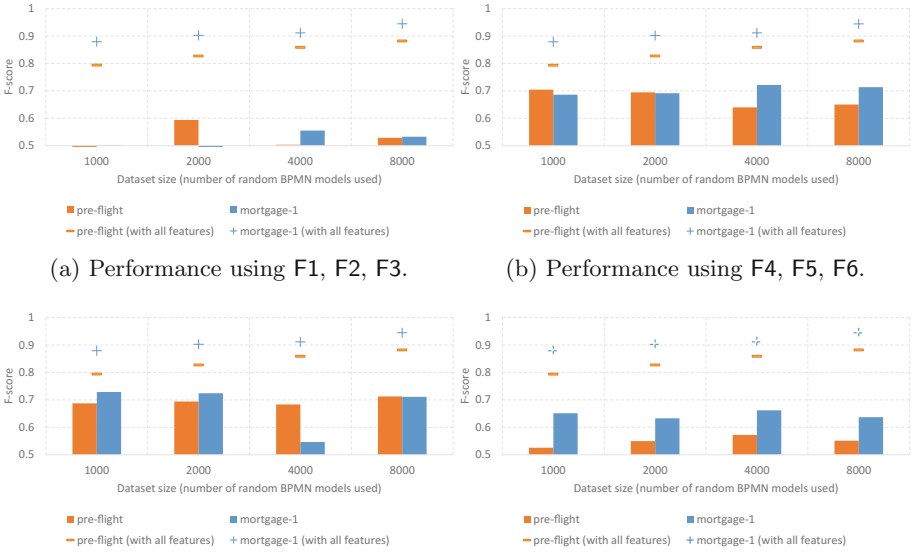


Fig. 3. F-scores on 10-fold cross validation tests, performed on different dataset sizes (models randomly selected), for the two tasks.

4.2 Prediction on Single Intermediate Models

The problem at hand is a binary classification problem (i.e., *novice* vs *expert*). Therefore, to characterize the quality of our predictions, we used the F-score (also known as F_1) measure. This measure is the harmonic mean of *precision* and *recall* and is suitable for capturing the classification quality [40]. We created different datasets with a different number of models (*up to* 1000, 2000, 4000 and 8000 BPMN models) considering BPMN models created during the last 70% of the modeling session (i.e., we discarded the intermediate models created during the initial 30% of the modeling session, to avoid almost-empty models). Then, using a 10-fold cross validation [26] we computed the average performance for each fold. Figure 3 depicts the outcomes of our tests. Clearly, the larger the dataset size, the better the outcome, since the system is able of better approximating the classification function. With the largest datasets, we were able to achieve an F-score of at least 0.88 (for pre-flight) and 0.94 (for mortgage-1). The comparison with 5 other classifiers is reported in the external appendix of this paper together with further implementation details (see footnote 1). To validate the necessity of our feature set, Fig. 4 shows, in turn, the F-scores, if subsets of features were used. The classification performance is notably worse when only subsets are used, as the classification model is not capable of accurately discriminating based just on these features. Additionally, the trend is not monotonically increasing but fluctuating, which suggests that the learning model is probably too complex given the data and it started to overfit the task, thus decreasing the performance. Furthermore, Fig. 5 shows the pairwise Pearson correlation coefficients for all features. Overall, there is little indication for linear correlations between the features, suggesting that they may indeed capture complementary aspects. To demonstrate that our approach (in line with requirement R3) is applicable in an online setting we conducted a performance analysis. The computation of the features was performed by instrumenting Cheetah to compute the features and measure the time. Only the time to compute the actual feature is taken into account. To compute the features, 18341 samples were randomly taken from the mortgage-1 dataset, which is the bigger of the datasets used. The test has been performed on a standard laptop with Windows 10 Enterprise and Java



(a) Performance using F1, F2, F3.

(b) Performance using F4, F5, F6.

(c) Performance using F7 and F8.

(d) Performance using F9 and F10.

Fig. 4. F-scores on 10-fold cross validation tests performed on different dataset sizes, considering different subsets of features. Each chart also reports, for comparison purposes, the values obtained using all features.

1.8, Processor Intel Core i7-7500U 2.7GHz and 16 GB RAM. During the test, a typical usage was maintained (i.e., no dedicated computation for the test, just to simulate a modeling environment, with several other software applications running at the same time). Figure 6 shows the average time required to compute each feature. Our results demonstrate that the calculation of most features is very fast. Feature F9, in turn, is more time consuming (93.12 ms). Still, all 10 features can be calculated in just a bit more than 100ms, which is certainly sufficient for application in the intended use cases.

4.3 Prediction of Modeler Expertise in the Entire Session

Our trained classifier predicts if model m_i was created by an expert if $exp(\boldsymbol{F}_i) = 1$ or by a novice if $exp(\boldsymbol{F}_i) = 0$ with an accuracy of 0.88–0.94. In this section we discuss how early $exp(\cdot)$ can predict that the modeler is an expert in a modeling session (from the features $\boldsymbol{F}_i$ of a partial model m_i).

To be robust against temporary changes in the classification over a few model snapshots, we derived a smoothing expert classifier $exp'(\boldsymbol{F}_i)$ that takes the average of $exp(\cdot)$ over the last N feature snapshots in the modeling sessions $\boldsymbol{F}_i, \boldsymbol{F}_{i-1}, \ldots, \boldsymbol{F}_{i-N+1}$. If $exp'(\boldsymbol{F}_i) > k$ for a threshold k, we consider m_i as an "expert model" in the session.

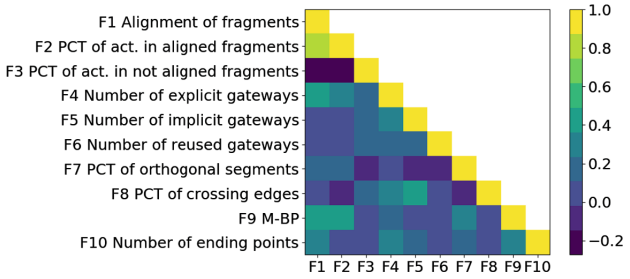


Fig. 5. Correlation coefficient matrix for all features.

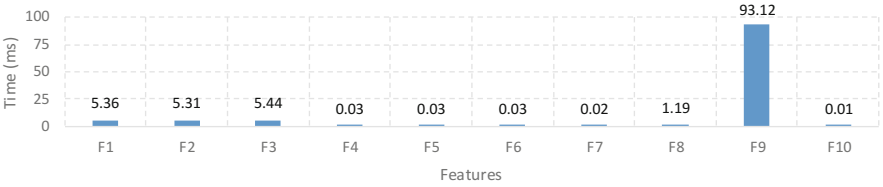


Fig. 6. Average time required to compute each feature over 18341 samples randomly selected from the mortgage-1 dataset (which is the biggest).

Using the $exp(\cdot)$ classifier trained on 8000 random samples from the last 70% of a modeling sessions (based on Fig. 4) and choosing $N = 20$ and $k = 0.6$ (based on hyper-parameter optimization) led to the following results: in the first 25% of any modeling session, $exp(\cdot)$ predicts any models to be from an "expert", i.e., *exp* errs on the "expert side". In the last 70% of a modeling session for mortgage-1 (pre-flight):

- exp' correctly predicted "expert" in 100% (94.5%) of the modeling sessions;
- in 90% (76.9%) of the sessions, this prediction occurred between 0.3-0.4 (0.3-0.55) of the modeling time at an average of 0.32 (0.35); exp' remained stable at "expert" until the end from 0.3-0.45 (0.3-0.75) of the modeling time onwards at an average of 0.33 (0.45);
- exp' falsely predicts "experts" for a short period in 13.8% (25.4%) of the modeling sessions of novices; the false prediction is stable until the end in just <1% (5%) of the novice sessions.

4.4 Discussion and Limitations

First and foremost, the presented approach to classify modelers meets the requirements R1-R4 put forward in Sect. 3.1. The classification is grounded in objective measures (R1), as the classification features capture properties of (intermediate) models only (e.g., the ratio of aligned model fragments or the type and usage of gateways). Feature calculation and classification from the model alone is fast (a bit more than 100 ms) rendering our approach unobtrusive (R2). Modelers do not have to spend any additional effort and, unless it is

desired, would not even be aware of the classification. Our experiments further highlighted that the features are well-suited to classify, potentially incomplete, intermediate models. We thus in addition conclude that the approach can be used online (cf. R3), coping with learning and improvement effects of modelers. Moreover, our approach is independent of a specific tool (cf. R4) as the respective features can be computed in any tool for the creation of activity-centric, flow-based process models with AND/XOR gateways.

Turning to the actual classification results, the general trend is encouraging, with an F-score of 0.88 for pre-flight and 0.94 for mortgage-1. Our results suggest that in terms of classification the mortgage-1 task seems easier when compared to the pre-flight one. This is plausible since the pre-flight lacks complex behavioral structures (i.e., no nested blocks or loops). Therefore, models of novices and experts do not differ so much as for more complex models.

Our results have impact both for modeling theory and practice. The ability to distinguish in an automated manner between groups of modelers (i.e., novices or experts) has potential applications in the context of teaching scenarios or as part of an adaptive modeling editor that classifies the user while modeling and adjusts itself based on the classification. The accuracy of the smoothing online-predictor exp' to distinguish novices and experts already early in the modeling session supports this idea; while reliable classification is only available after a third of the modeling session, it may be exactly at the right time to identify novices and offer support. The false positive rate of up to 25% in temporary expert classifications suggests that users need to stay in control of adaptations of the modeling environment.

The assessment of expertise and professional capabilities is needed for different purposes: for recruitment, for deciding on assignment of employees to tasks, for team formation [41], or for forming relatively uniform groups for training. Current approaches for this assessment (e.g., based on success in a modeling task) suffer limitations – specifically biases that stem from differences in domain knowledge, from the specific modeling task selected for the assessment, or from accidental success or failure. The approach we present overcomes these limitations by considering a combination of features which are evidence-based and less subject to conscious and intentional manipulation: it would be very difficult to intentionally introduce bias in such assessment.

Although the feature extraction was conducted in the context of Cheetah platform, it is not dependent on a specific modeling tool, but can be generalized to any other BPMN-based process modeling environment. The general approach could also be applied to modeling notations other than BPMN, however, in this case the feature extraction would need to be adapted to the specific notation.

The features F1-F10 considered in this paper *together* are relevant for accurately classifying expert and novice modelers: sub-groups of features show significantly lower F-scores than all features combined (Fig. 4), and the features discriminate even partial expert and novice models (Sect. 4.3). Other classifiers can be used for the same set of features with the same or even better accuracy (see footnote 1). A threat to generalizability to larger models and applicability

in individual cases may be the thresholds used in F1–F3 and F7. Validating the features against more (and larger models) and modeler preferences is subject of future work.

Another limitation of our work is that we used the same tasks for training and prediction. This is a setting that is applicable to teaching or recruitment scenarios, where many modelers work on the same modeling task. To improve the generalizability of the approach and to make it applicable in settings where models are more heterogeneous, inter-model predictions are required. Some of the features considered in our prediction depend on the model (e.g., the number of gateways) and thus have to be adapted for inter-model predictions.

5 Conclusion and Future Work

In this paper we demonstrated that novices and experts can be differentiated to a large extent based on how they lay out their model. By basing our classification approach on a model's layout, we were able to provide an approach that is automated, based on objective-measures, that is unobtrusive, and independent of a particular modeling tool. Our performance analysis further demonstrated that the approach is applicable in online settings. With this paper we focused on inter-model classification.

Future work will generalize the approach by additionally considering intra-model classification. We plan to continue the feature engineering and selection processes by considering features that capture the evolution of model properties over time. In parallel, different prediction techniques can also be investigated.

Acknowledgements. This research was funded by the Austrian Science Fund (FWF): P26140–N15 and P26609N15.

References

1. Burton-Jones, A., Meso, P.: The effects of decomposition quality and multiple forms of information on novices' understanding of a domain from a conceptual model. J. AIS **9**(12), 748–802 (2008)
2. Fettke, P.: How conceptual modeling is used. Commun. AIS (CAIS) **25**, 571–592 (2009)
3. Recker, J., Safrudin, N., Rosemann, M.: How novices design business processes. Inf. Syst. **37**(6), 557–573 (2012)
4. Soffer, P., Kaner, M., Wand, Y.: Towards understanding the process of process modeling: theoretical and empirical considerations. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM 2011. LNBIP, vol. 99, pp. 357–369. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28108-2_35
5. Wickens, C.D., Hollands, J.G.: Engineering Psychology and Human Performance, 3rd edn. Pearson, London (1999)
6. Figl, K.: Comprehension of procedural visual business process models a literature review. Bus. Inf. Syst. Eng. **59**, 41–67 (2017)
7. Koschmider, A., Reijers, H.A.: Improving the process of process modelling by the use of domain process patterns. Enterp. IS **9**(1), 29–57 (2015)

8. Koschmider, A., Hornung, T., Oberweis, A.: Recommendation-based editor for business process modeling. Data Knowl. Eng. **70**(6), 483–503 (2011)
9. Weber, B., Reichert, M., Rinderle-Ma, S.: Change patterns and change support features - enhancing flexibility in process-aware information systems. Data Knowl. Eng. **66**(3), 438–466 (2008)
10. Gschwind, T., Koehler, J., Wong, J.: Applying patterns during business process modeling. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 4–19. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85758-7_4
11. Claes, J., Vanderfeesten, I.T.P., Gailly, F., Grefen, P., Poels, G.: The structured process modeling method (SPMM) what is the best way for me to construct a process model? Decis. Support Syst. **100**, 57–76 (2017)
12. Claes, J., Vanderfeesten, I., Pinggera, J., Reijers, H.A., Weber, B., Poels, G.: Visualizing the Process of process modeling with PPMCharts. In: La Rosa, M., Soffer, P. (eds.) BPM 2012. LNBIP, vol. 132, pp. 744–755. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36285-9_75
13. Pinggera, J., et al.: Styles in business process modeling: an exploration and a model. Softw. Syst. Model. **14**, 1055–1080 (2013)
14. Krogstie, J.: Quality of models. In: Krogstie, J. (ed.) Model-Based Development and Evolution of Information Systems, pp. 205–247. Springer, London (2012). https://doi.org/10.1007/978-1-4471-2936-3_4
15. Mendling, J., Recker, J.C., Reijers, H., Leopold, H.: An empirical review of the connection between model viewer characteristics and the comprehension of conceptual process models. Inf. Syst. Front., 1–25 (2018)
16. Larkin, J., McDermott, J., Simon, D.P., Simon, H.A.: Expert and novice performance in solving physics problems. Science **208**(4450), 1335–1342 (1980)
17. Reijers, H.A., Mendling, J.: A study into the factors that influence the understandability of business process models. IEEE Trans. Syst. Man Cybern. - Part A: Syst. Hum. **41**(3), 449–462 (2011)
18. Petre, M.: Why looking isn't always seeing: readership skills and graphical programming. Commun. ACM **38**(6), 33–44 (1995)
19. Batra, D., Davis, J.G.: Conceptual data modelling in database design: similarities and differences between expert and novice designers. Int. J. Man Mach. Stud. **37**(1), 83–101 (1992)
20. Narasimha, B., Leung, F.S.: Assisting novice analysts in developing quality conceptual models with UML. Commun. ACM **49**(7), 108–112 (2006)
21. Jawaheer, G., Weller, P., Kostkova, P.: Modeling user preferences in recommender systems: a classification framework for explicit and implicit user feedback. ACM Trans. Interact. Intell. Syst. **4**(2) (2014). Article no. 8
22. Riedl, R., Léger, P.-M.: Fundamentals of NeuroIS-Information Systems and the Brain. SNPBE. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-45091-8
23. Crk, I., Kluthe, T., Stefik, A.: Understanding programming expertise: an empirical study of phasic brain wave changes. ACM Trans. Comput.-Hum. Interact. **23**(1), 2:1–2:29 (2016)
24. Pinggera, J.: The process of process modeling. Ph.D. thesis, University of Innsbruck (2014)
25. Martini, M., Pinggera, J., Neurauter, M., Sachse, P., Furtner, M.R., Weber, B.: The impact of working memory and the process of process modelling on model quality: investigating experienced versus inexperienced modellers. Sci. Rep. **6** (2016). Article no. 25561

26. Aggarwal, C.C.: Data Mining. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-14142-8
27. Baker, R.: Big Data and Education. Columbia University, New York (2015)
28. Mendling, J., Reijers, H.A., van der Aalst, W.M.P.: Seven process modeling guidelines (7PMG). Inf. Softw. Technol. **52**(2), 127–136 (2010)
29. Polyvyanyy, A.: Structuring process models. Ph.D. thesis, University of Potsdam (2012)
30. Haisjackl, C., Soffer, P., Lim, S.Y., Weber, B.: How do humans inspect BPMN models: an exploratory study. Softw. Syst. Model. **17**, 655–673 (2016)
31. Bernstein, V., Soffer, P.: Identifying and quantifying visual layout features of business process models. In: Gaaloul, K., Schmidt, R., Nurcan, S., Guerreiro, S., Ma, Q. (eds.) CAISE 2015. LNBIP, vol. 214, pp. 200–213. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19237-6_13
32. Gschwind, T., Pinggera, J., Zugal, S., Reijers, H.A., Weber, B.: A linear time layout algorithm for business process models. JVLC **25**(2), 117–132 (2014)
33. Figl, K., Strembeck, M.: On the importance of flow direction in business process models. In: Proceedings of ICSOFT-EA, pp. 132–136 (2014)
34. Burattin, A., Bernstein, V., Neurauter, M., Soffer, P., Weber, B.: Detection and quantification of flow consistency in business process models. SoSyM **17**(2), 633–654 (2017)
35. Mitchell, T.M.: Machine Learning. McGraw-Hill, New York City (1997)
36. Hornik, K.: Approximation capabilities of multilayer feedforward networks. Neural Netw. **4**(2), 251–257 (1991)
37. Huang, G.B.: Learning capability and storage capacity of two-hidden-layer feedforward networks. IEEE Trans. Neural Netw. **14**(2), 274–281 (2003)
38. Hagan, M., Demuth, H., Beale, M., De Jesús, O.: Neural Network Design (2014). Oklahoma
39. Pinggera, J., Zugal, S., Weber, B.: Investigating the process of process modeling with cheetah experimental platform. In: Proceedings of the ER-POIS, pp. 13–15 (2010)
40. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval, 1st edn. Cambridge University Press, Cambridge (2008)
41. Niknafs, A., Berry, D.: The impact of domain knowledge on the effectiveness of requirements engineering activities. Empir. Softw. Eng. **22**(1), 80–133 (2017)