Process Mining to Unleash Variability Management: Discovering Configuration Workflows Using Logs

Ángel Jesús Varela-Vaca, José A. Galindo, Belén Ramos-Gutiérrez
María Teresa Gómez-López and David Benavides
Universidad de Sevilla
Seville, Spain
{ajvarela,jagalindo,brgutierrez,maytegomez,benavides}@us.es

ABSTRACT

Variability models are used to build configurators. Configurators are programs that guide users through the configuration process to reach a desired configuration that fulfils user requirements. The same variability model can be used to design different configurators employing different techniques. One of the elements that can change in a configurator is the configuration workflow, i.e., the order and sequence in which the different configuration elements are presented to the configuration stakeholders. When developing a configurator, a challenge is to decide the configuration workflow that better suites stakeholders according to previous configurations. For example, when configuring a Linux distribution, the configuration process start by choosing the network or the graphic card, and then other packages with respect to a given sequence. In this paper, we present COnfiguration workfLOw proceSS mIning (COLOSSI), an automated technique that given a set of logs of previous configurations and a variability model can automatically assist to determine the configuration workflow that better fits the configuration logs generated by user activities. The technique is based on process discovery, commonly used in the process mining area, with an adaptation to configuration contexts. Our proposal is validated using existing data from an ERP configuration environment showing its feasibility. Furthermore, we open the door to new applications of process mining techniques in different areas of software product line engineering.

KEYWORDS

variability,
configuration workflow,
process mining,
process discovery,
clustering

1 INTRODUCTION

Variability models such as Feature Models (FMs) [22] describe commonalities and variabilities in Software Product Lines (SPLs) and are used along all the SPL development process. After an FM is defined, products can be configured and derived. In the configuration and derivation process, users select and deselect features using a *configurator*. A configurator [21][19] is a software tool that presents configuration options to the users in different stages. An example of a configurator tool is KConfig [58] where developers can configure the Linux kernel with more than 12.000 configuration options.

An important aspect of a configurator is to determine the *configuration workflow* [28], i.e., the order in which features and options are presented to configuration stakeholders. For instance, when configuring the Linux kernel using KConfig [58], there can be different user configuration profiles depending on interests or skills. The configuration workflow used by a configurator can impact the user experience in the configuration process. Therefore, selecting a well suited configuration workflow is a challenge. Up to now – to the best of our knowledge– the selection of a configuration workflow is made either intuitively or following the structure and properties of a variability model [21, 66].

In this paper, we present COLOSSI, an approach that takes a feature model and a set of existing configuration logs and automatically retrieves configuration workflows. A configuration log is a set of configurations performed in the past in a given domain taking into account a configuration order. Our solution relies on *process mining* [3] techniques. Process mining is a well established area of business process management that uses different techniques to extract business processes from traces of execution. In our approach, we conceptually map a business process model to a configuration workflow and traces to configuration logs making possible to reuse process mining techniques to infer configuration workflows.

Although using process mining can automatically retrieve configuration workflows, the results can be difficult to interpret to domain engineers in order to build a configurator. This is mainly due to the fact that, very often, mined processes are "spaghetti-like" models in which the same activity needs to be duplicated [62]. To illustrate the difficulty, Figure 1 shows the result of directly applying process mining techniques to the ERP system presented in [46] and detailed in Section 3.

The simplification of spaghetti processes is an open problem in the process mining domain [3]. Variability models have special characteristics that can help to guide the discovery process. To overcome this difficulty, our solution adapts a clustering algorithm

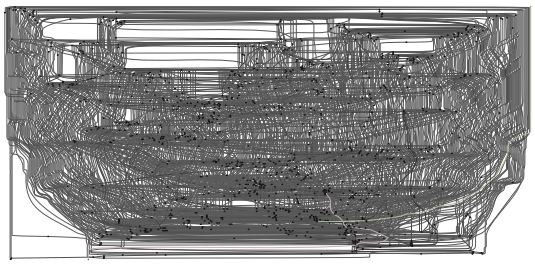

Figure 1: Spaghetti process of the ERP presented in [46].

solution that instead of retrieving a single–complicated configuration workflow, is able to cluster the configurations according to different metrics. Our solution takes information from the variability model as input and retrieves less complex configuration workflows that can assist the development of better configurators.

COLOSSI is validated using an ERP case study taken from [46]. Results show that the metrics of the retrieved configuration workflows are improved by a 99% in the best case and 81% in the worst case with respect to the spaghetti like first solution.

The contributions of this paper are as follows:

- An automated technique based on process mining to select a configuration workflow that better fits the configuration logs according to a set of metrics.
- A validation of the proposal using a realistic ERP scenario.
- An available implementation that can be applied to other datasets.

The remainder of this paper is organised as follows: Section 2 details the solution and concepts that grounds our proposal; Section 3 presents empirical results from analysing COLOSSI; Section 4 presents the related work and Section 5 presents concluding remarks and lessons learned.

2 COLOSSI: CONFIGURATION WORKFLOW PROCESS MINING SOLUTION

In order to create a configuration workflow, a feature model and a configuration log must be combined. Figure 2 shows an overview of the COLOSSI approach. Using the configuration log, it is possible to apply process mining techniques to derive a valid configuration workflow representing all the possible paths defined in the configuration logs. It is likely that the resulting workflow follows the so-called spaghetti-style [62] and it is therefore difficult to understand and manipulate. Nevertheless, it is important to remark that it can be already exploited by process mining automated tools to extract metrics, perform simplification over the workflow as well as many additional analysis. Also, any generated configuration workflow can be already used to build automatically a configurator.

In addition, to the usage of process mining techniques, we propose handling and clustering methods to reduce and group similar configuration traces according to some properties. Those clusters can then be used again as input of process mining techniques to obtaining a set of configuration workflows depending on the observed behaviour of the configuration logs. Those workflows will obtain better metrics with respect to the original complex workflows of step ①. Our conjecture is that the resulting configuration workflows of step ② will better guide the domain engineers in the construction of a configurator as well as the analysis mentioned previously.

Following, we describe the details of the different elements of COLOSSI.

2.1 Inputs

A feature model is an arranged set of features that describes variability and commonality using features and relationships among them. [18, 57]. FMs describe all the potential combinations of features. Figure 3 shows an excerpt of a feature model of the ERP domain where features are arranged in a tree–like structure and different relationships are established among them. FMs can be used to build configurators that are pieces of software that guide the configuration process while selecting and deselecting features. An example of a configurator is KConfig, a tool that helps configuring the Linux kernel. As an FM can define a configuration space defined by all the possible feature combinations, it can also define different possible configuration workflows that can be derived using the same FM.

COLOSSI takes as input a FM and a *configuration log*. To define a configuration log, we use some definitions that are used in process mining area to define events and traces and we map those definitions to define a configuration log.

An event log is a multiset of traces:

Definition 2.1. (Event Log). Let L be an event log $L = [\tau_1, \cdots, \tau_m]$ as a multiset of traces τ_i.

A trace is a tuple with an identifier and a sequence of events that occurred at some point in time:

Definition 2.2. (Trace). Let τ be a trace $\tau = \langle case\_id, \mathcal{E} \rangle$ which consists of a *case_id* which identifies the case, and a sequence of events $\mathcal{E} = \{\varepsilon_1, \cdots, \varepsilon_n\}$, ε_i occurring at a time index i relative to the other events in $\mathcal{E}$.

An event occurrence is a 3–tuple with an identifier of an activity that occurred at some timestamp and that can have additional information:

Definition 2.3. (Event occurrence). Let ε be an event occurrence $\varepsilon = \langle activity\_id, timestamps, others \rangle$ which is specified by the identity of an activity which produces it and the timestamps. It can store more information (i.e., states, labels, resources, etc.)

In COLOSSI, we conceptually map elements from the feature modelling domain to the process mining domain as shown in Table 1. Concretely, an event log is conceptually a configuration log. A trace is an ordered configuration, i.e., a *configuration trace*, thus, it is a set of selected features that follow a given order. Finally, an event occurrence is a feature. Additionally, a feature can have more information like attributes associated with this feature such as preferences, metrics or the like.

2.2 Configuration logs extractor

A configuration log is composed of a set of configuration traces where each configuration trace encodes not only the features of a configuration but the timestamps indicating when each feature

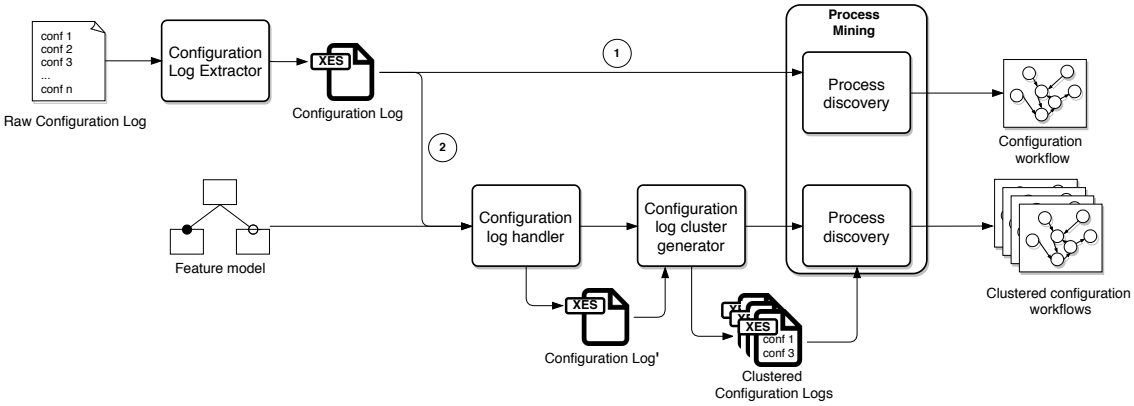

Figure 2: COLOSSI solution overview.

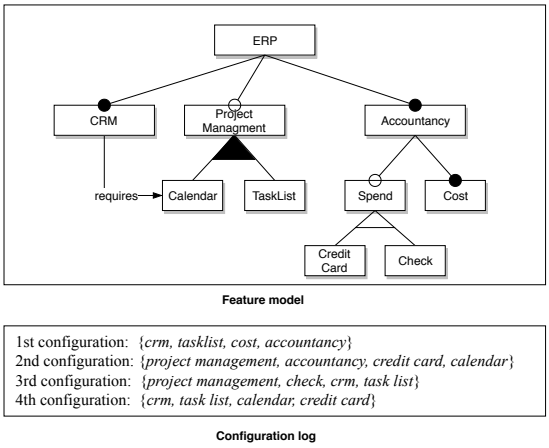

Figure 3: ERP domain based example.

Process Mining	Product Line
Event log	Configuration log
Trace	Configuration trace
Event occurrence	Feature

Table 1: Mapping concepts.

was selected. In a raw configuration log, we can find a diversity of meta-information among the selected or deselected features. Moreover, this meta-information can be presented in a unstructured or structured fashion.

In this first step, we take as input a raw configuration log and output a set of configuration traces. Therefore, we need to *i*) search for the meta-information encoding the timestamps for each feature. Note that this might not be explicit and can be provided using other mechanisms (e.g., line numbers in a plain text format); *ii*) use this meta-information to represent the feature selection order, and; *iii*) store the set of configuration traces in a format that can be used throughout the configuration workflow retrieval process (e.g., XES serialization [1]). After this, we end up with a set of configuration traces that represent the selection order used by the configurator users. However, there might be non-valid configurations and other erroneous configurations w.r.t domain information.

2.3 Configuration logs handler

At this step, the configuration log might contain non-valid configurations, erroneous partial selection of features among other domain-related errors such as those depicted in [6]. To remove clutter and noise out of the workflows, users might prefer to remove such information from the configuration log. This cleaning step consist on removing wrong selection of features (a.k.a non-valid partial configurations) as well as generate metrics that can be latter exploited to optimise the workflow retrieval process. For example, the use of atomic-sets to complete partial configurations.

Depending on the expected workflow usage, domain engineers have to define the meaning of a valid configuration and the metrics to rely on. For example, a SPL engineer might consider only configurations with complete assignments of features to develop a configuration while other might find interesting to consider full assignments (i.e., to configure only the variability part of the product line, keeping aside the common parts).

2.4 Process mining - discovery

Process mining is a family of techniques based on event logs that can be categorised as process discovery, conformance checking and enhancement [63]. In this paper, we are focused on the use of *process mining* to analyse the configuration logs for the discovering of configuration workflows based on the user experiences. Process discovery in process mining brings together a set of algorithms to generate a workflow process model that covers the traces of activities observed in an organisation [40]. The evolution of algorithms during last decades has allowed the discovery of complex models that are able to involve not only the activities executed in the daily work of companies, but also the persons who execute them and the used resources.

Process mining is an important topic that has been well received by the enterprises, bringing about the evolution of the research solution tools (e.g., ProM [64]) to commercial solutions (e.g., Disco™and Celonis™). This facilitates its applicability to several contexts and areas, although variability has been out of the scope of these techniques before this paper.

Process discovery in process mining uses a set of traces similar to the configuration log shown in Figure 3, to obtain a model that

covers the possible traces. Figure 4 shows the process discovered by Disco tool-suite, which covers every possibility configuration trace. The relational patterns among the definition of the features become part of the model. For example, two features can be the first in the traces (*CRM* or *Project management*) or after *CRM* always *Task List* is selected. Figure 4 also shows the number of traces that are represented by each transition, giving information about the importance o each part of the traces in the obtained model.

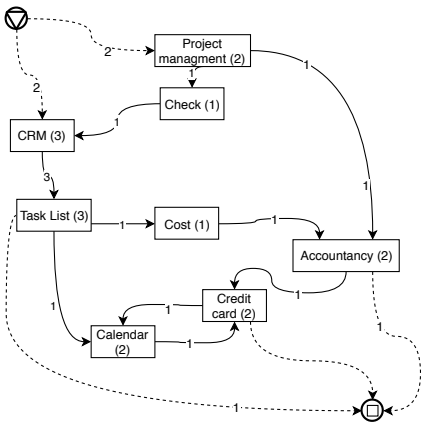

Figure 4: Process discovered for configuration log of ERP domain based example.

In the framework proposed in this paper (Figure 2), *Process mining - process discovery* module enables to read an event log and generates a process model that fits these traces. In the case of the variability context, a *configuration log* is read and a *configuration workflow* is obtained using the same techniques used for classical process mining.

2.5 Configuration logs cluster generator

Configuration processes have a high degree of variability, specially when the configuration order is defined by human decisions. The application of process discovery in this type of scenarios tends to produce spaghetti-like processes, being necessary to apply a pre-processing. Configurability contexts are specially variable in relation to the executed activities derived from the high human intervention, thereby, we propose to divide the traces into subsets, to model different profiles of users and avoiding the discovery of non-user understandable processes. In these contexts, where process discovery is used to infer spaghetti-like processes, frequently clustering techniques such as a pre-processing step are applied [26]. To adapt the solution to configuration tasks, we propose the division of the configuration traces into multiple clusters before the application of a process discovery. This division lets to discover configuration workflows with more quality. This section describes what a cluster is and the metric (e.g., entropy) used to divide the traces among them. In following sections, we describe how quality is measured and how the clusters can be created.

Being L a configuration log composed of a set of configuration traces (i.e., $[\tau_1, \cdots, \tau_m]$), a cluster is a subset of configuration traces from L that complies certain properties.

Definition 2.4. (Cluster of Configuration Traces). A cluster of configuration logs, $c = [\tau_i, \cdots, \tau_j] \subseteq L$, where $\forall \tau_k \subseteq L, \exists c \mid \tau_k \in$ c and $\nexists$ c' $\neq$ c where $\tau_k \in$ c'.

The distribution of configuration traces between various clusters depends on the purpose of the practitioners. In our case, the goal is to group the more similar configuration traces. In this paper, the meaning of 'similar' is related to both features and transitions involved in the logs. For this reason, we adapted the classical information entropy metric [37] by introducing two different custom entropy metrics for clustering in the configuration context:

- *Entropy-features ($S_{features}$)* of a cluster: a metric which measures the similarity between a set of traces according to the features that belong to the same cluster. Thus, it is the ratio between the number of features that do not appear in all configuration traces ($features_{nat}$) and the number of different features in all the configuration traces ($features_{diff}$):

$$S_{features} = \frac{|features_{nat}|}{|features_{diff}|} \tag{1}$$

- *Entropy-transitions ($S_{transitions}$)* of a cluster: a metric which measures the similarity between a set of traces according to the transitions that belong to the same cluster. Thus, it is the ratio between the transitions that do not appear in all configuration traces ($transitions_{nat}$) and the number of different transitions in all the configuration traces ($transitions_{diff}$):

$$S_{transitions} = \frac{|transitions_{nat}|}{|transitions_{diff}|} \tag{2}$$

In order to illustrate the calculation of entropies, the $S_{features}$ and $S_{transitions}$ for the *Cluster 1* and *Cluster 2* of Figure 5 are determined in Table 2.

	Entropy-features	*Entropy- transitions*
Cluster 1	$\frac{0}{4} = 0$	$\frac{0}{3} = 0$
Cluster 2	$\frac{6}{8} = 0,75$	$\frac{6}{15} = 0,4$

Table 2: Entropies for the clusters of the Figure 5.

Note that the range of the entropy is [0..1]. The values of entropy that are close to 0 represent more similar traces, whilst when they are close to 1 represent that there are different features involved in the traces of the cluster. The best configuration of clusters obtained from a set of configurations traces is the one that minimize the summation of the entropy of all clusters obtained. The challenge is how to obtain the best configuration of clusters as a pre-processing of a process discovery.

In order to find out the best configuration traces divided into clusters, minimising the entropy of the resulting clusters, different algorithms for clustering can be used. In accordance with [30] *clustering* provides an unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters). *Clustering* [31] brings together a large set of algorithms that can be classified in different ways according to the point of view necessary in the case of study. We propose the use of the well-known *hierarchical agglomerative clustering* algorithm base on the work in [69] as the combination of hierarchical and agglomerative clustering.

On the first hand, *hierarchical clustering* is defined as a procedure to form hierarchical groups of mutually exclusive subsets, each of which has members that are maximally similar with respect to the specified characteristics [69]. In the same study, authors define the process as: assuming we start from n sets, it permits their reduction to $n-1$ mutually exclusive sets by considering the union of all possible $\frac{n(n-1)}{2}$ pairs and selecting a union having a maximal value for the objective function.

On the other hand, *agglomerative clustering* is an algorithm that starts from the assumption that each element constitutes a cluster by itself (singleton) and it successively merges these singletons together forming clusters until a stopping criterion is satisfied which is also determined by the objective function.

The characteristics used in our solution is based on both entropies presented (features and transitions), combined with the objective function the Ward's minimum variance method [69]. This function aims to minimise the sum of the squared differences within all clusters, which means, a variance-minimising approach. Figure 5 shows the obtained dendrogram[1] for the example in Figure 3 by using Entropy-features.

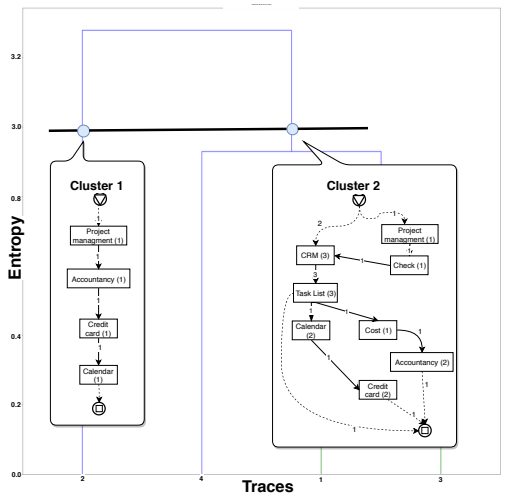

Figure 5: Clustering for the ERP excerpt using the *Entropy-features*.

It is well-known that every clustering algorithm builds a distance matrix during its execution based on a given methodology (euclidean, manhattan, etc.). However, *hierarchical agglomerative clustering* is an exception, since it can be carried out from the distance matrix itself. We consider the entropy matrix as the distance matrix, this leads us to decide for this method of clustering, which can be performed also using other methods, such as *single-linkage*, *complete-linkage*, *average-linkage*, and *Ward*.

Whenever a clustering process is executed, one of the first problems to deal with is to decide which is the optimal number of clusters. Many studies carried out about the discovery of the optimum number of clusters, therefore, we decided to study a significant number of them to choose by voting the number of clusters that most indicators had selected as optimal. In this regard, 17 different indicators [4, 5, 9, 14, 16, 17, 20, 24, 25, 29, 32, 33, 42, 44, 45, 50, 51] are used as reference to choose the best number of clusters adapted to our scenario.

By using these indicators with the different clustering methods mentioned above, all of them selected a too large number of clusters as the optimal solution. In addition, the values of the indicators themselves are in some of the cases too dissimilar.

For all the methods the range [0-10] is proposed to determine the number of clusters. In the *single-linkage* method always retrieved the maximum (i.e., 10). For the *average-linkage* retrieved always an average of 2 clusters. Both methods help to bound the limits of number of clusters between 2 and 10. Only with the *complete-linkage* and *Ward* methods, more homogeneous, similar and an assumable number of clusters as results were obtained for the dendrograms showed in Figures 6 and 7.

Finally, by making use of dendrograms, it is observed that applying the *Ward* method, the samples becomes better distributed among the clusters, thereby, generating more differentiated clusters and better structured dendrograms. This fact determines the *Ward's* method as the best option for our approach.

2.6 Leveraging the results of COLOSSI

The COLOSSI approach can be used in different scenarios to leverage process mining in variability management. One of the scenarios presented in this paper is the building of configurators. However, we envision other areas where process mining can be used to automate different tasks. Next, we describe those scenarios, also related to software product lines, from our experience and perspective:

- **Configurator building**. Up to now, configurators building is performed using manual mechanisms or, at most, using the information present in the variability model (e.g., tree traversal in feature models). With COLOSSI, we open the door to use existing configuration logs to build configurators. This novel approach can open the door to new ways of assisting configurators builders by using the generated configuration workflow to optimise configurators.
- **Data analysis**. From the generated configuration workflow it is possible to perform many analysis in terms of graph metrics. Deadlocks identifications, misalignment analysis, metrics extraction –to just mention a few– are areas where process mining techniques can be useful.
- **Testing**. From the data extracted in the former item, it could be possible to define new sampling techniques [61] that can improve the identification of bugs or feature interactions in existing product lines.
- **Variability reduction**. One of the challenges for companies that develop software product lines is variability reduction [8]. While variability is a must in a software product line approach, it is always difficult to find a trade off between a high degree of variability and a systematic management of such a variability. In this context, experts claim for techniques and tools to reduce variability while preserving configurability. Process mining techniques presented in this paper can be a

[1] *Dendrogram* is a branching diagram which represents the arrangement of the clusters produced by the corresponding analyses

first step towards defining tools to assist in the decision of variability reduction.

- **Reverse engineering**. One of the inputs used when reverse engineering feature models are configurations (a.k.a. product matrix). We envision that the techniques described in this paper can be used in reverse engineering of variability models. For instance, the generated configuration workflow can be analysed to better guide reverse engineering algorithms

3 EVALUATION

In this section we present the evaluation of COLOSSI. The evaluation consists of the application of the framework detailed in Section 2 to a configuration log obtained from a real scenario. The possible clusters derived from the application of the defined entropies are analysed.

3.1 Experimentation data

In order to analyse the applicability of our example in a configuration real scenario, we used the raw information from [46]. The used data include a configuration model representing a real ERP, as well as a raw configuration log. The ERP feature model has 1920 features and 59044 cross-tree constraints. Also, the configuration log is formed of 35193 event occurrences that represent a total of 170 different configuration traces with an average of 207 features per configuration trace.

3.2 Framework application

In this section we detail each task of the framework presented in Figure 2.

3.2.1 Configuration Log Extractor. The input data of the configuration of the ERP is represented in a CSV file with two elements, the configuration id and the feature that is configured. Note that a feature can appear in one or more traces, but no more than once in the same trace. Then, the timestamp required to extract the traces was taken by the line number in which the features where appearing throughout the file in a sequential order. This is, we assume that the timestamps were implicit based on the order of appearance (i.e., line numbers) then, transformed them into a more standard format for traces. Concretely we use in our solution the IEEE Standard for eXtensible Event Stream (XES) [1]. This is a standard to serialise, store, exchange events data and it is commonly used in process mining techniques.

3.2.2 Configuration Log Handler. To clean up the set of configurations retrieved by the extractor we decided to consider only valid partial and full configurations. This filtering operation is performed by using the FaMa framework [7]. After filtering, the valid partial configurations using automated analysis [6], we ended up considering 61 configuration traces from the initial set of 170.

3.2.3 Configuration Log Cluster Generator and Process Discovery. Figure 6 and 7 represent the obtained clusters according to the dendrogram built by means of the entropy of features and transitions analysis respectively. In the case of feature entropy (Figure 6), five clusters are obtained. On the other hand, when the transition entropy is used, three clusters are derived to split the configuration logs into simpler configuration workflows. In the following subsections, the details about the obtained configuration workflows and clusters are analysed.

3.3 Analysis of Results

In order to evaluate how the application of clustering can improve the configuration workflows obtained by COLOSSI, in this section, we compare the models discovered: (1) the *original* configuration logs obtained from the initial raw configuration log (cf. Section 3.1); (2) the *filtered* version of the same log including only valid configurations (i.e., after applying *configuration log handler*), and; (3) two set of clusters based on the proposed entropies (features and transitions explained in Section 2.5).

The analysis is carried out following two different perspectives: (1) the analysis of the discovered configuration workflows and (2) the analysis of the set of configuration traces involved in each cluster used in the process discovery.

3.3.1 Analysis of discovered configuration workflows. First, highlight that inductive process discovery techniques used by COLOSSI, ensure the soundness and correctness of the process models obtained [35]. Thus, an analysis of the soundness and correctness of the configuration workflows are unnecessary since processes discovered is always complete, have a proper completion, and have no dead transitions.

However, the complexity of the configuration models is affected by the number of features, the number of configuration traces and the number of transitions. Obviously, the filtering of the configuration traces or the division of the logs will bring about simpler configuration workflows. Figure 8 depicts in a comparative way the number of features, configuration traces and transitions of the set of configuration logs using in each scenario: original configuration log, filtered configuration log only with valid traces, the 5 clusters obtained by using entropy-features, and the 3 clusters obtained by using entropy-transitions.

Regarding general parameters, there is an enormous difference between the original, filtered version, and both that use clustering. The original configuration workflow contains 1652 features and 3330 transitions, whilst the filtered version has one less magnitude order of features and transitions. The clusters seem similar to the filtered version, however, each cluster contains, at least, less than half regarding features. In case of the transitions, the filtered version reached four times fewer transitions than the original. However, clusters reached in the worst case 500 transitions less than the filtered and more than one third fewer transitions in the best case. Due to the clusters group a set of similar traces, the number of traces is intrinsically smaller regarding the configuration traces of the original and filtered configuration workflow.

In conclusion, clusters enable to reduce the complexity of configuration workflow discovered by reducing the configuration traces involved in the same configuration workflow. However, the question is what level the quality of the obtained workflow is improved, and which distribution of cluster-entropy works better.

In literature, several metrics are used to measure how "good" is a design of a business process model [10, 43, 48]. Discovered configuration workflows are also processes with features instead of activities, therefore, these metrics can be adapted to measure

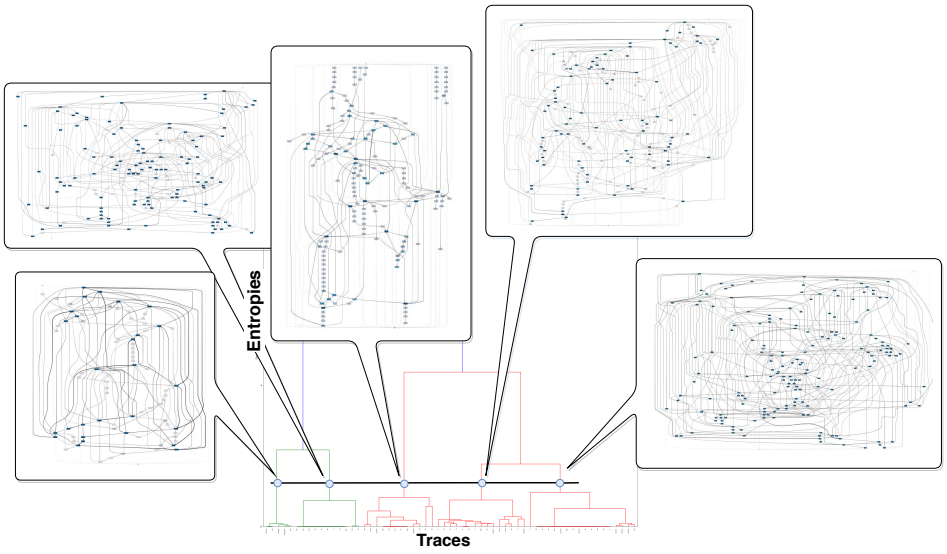

Figure 6: Clustering for the ERP example using the *Entropy-features*.

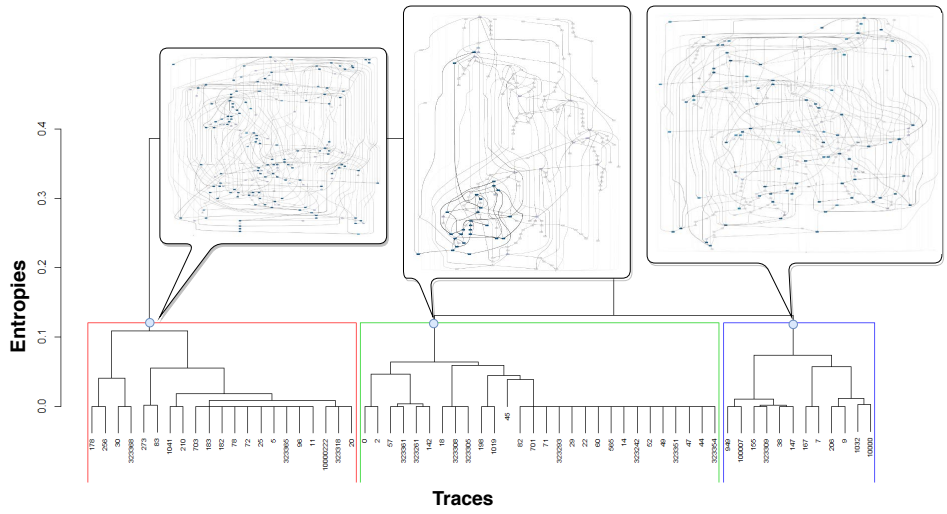

Figure 7: Clustering for the ERP example using the *Entropy-transitions*.

the quality of our obtained configuration workflows. The next set of metrics is adapted to measure the understandability and the complexity of the configuration workflows to compare the four discovered configuration workflows:

- *Density*: the ratio of transitions divided by the maximum number of possible transitions. The lower the value of density, the higher the understandability and the lower complexity.
- *Cyclomatic number* (*CC*): the number of paths needed to visit all features. The cyclomatic can be seen as a complexity metric, thus, the lower the value of *CC*, the lower the level of complexity.
- *Coefficient of connectivity* (*CNC*): the ratio of transitions to features. The greater the value of *CNC*, the greater the complexity of configuration workflows. Although, the authors in [43] remark that models with the same *CNC* value might differ in complexity regarding this parameter.
- *Control Flow Complexity* (*CFC*) enables to measure the complexity in terms of the potential transitions after a split depending on its type. The greater the value of the *CFC*, the greater the overall structural complexity of a workflow.

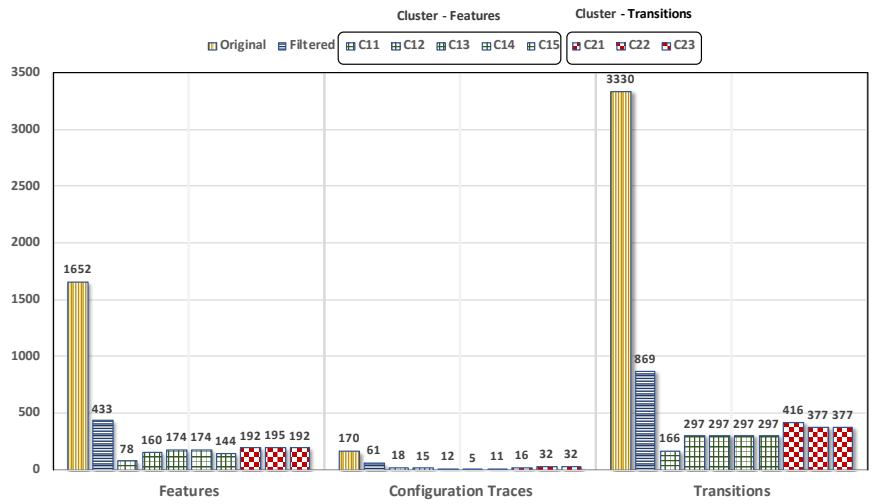

Figure 8: Characteristics of the configurations logs.

Table 3 shows the results of these metrics obtained from the discovered configuration workflows. As explained, for every metric the lower value they take, the better the understandability and less complexity. We shall highlight that the metrics associated with the different clusters are aggregated as arithmetic means for a better comparison.

Config. Workflow	*Density*	*CC*	*CNC*	*CFC*
Original	0,00123	1679	2,016	1677
Filtered	0,00464	437	2,069	434
Cluster-Features	0,01469	125,8	1,892	118,4
Cluster-Transitions	0,00632	118,8	1,213	156

Table 3: Metrics of the configuration workflows.

It is interesting to note how configuration workflow for the original has the lowest density in comparison with the others. This is because the number of features is so high and it compensates the largest number of transitions. However, it has the highest complexity compared to *CC* and *CFC*. However, no conclusions can be achieved with regard to the complexity based on *CNC*. Therefore, although the density is the lowest, the other three metrics demonstrate that the workflow for the original is very complex and misunderstood. On the other hand, the workflow for the filtered version has the highest density, thus, it is the less understandable regarding to this metric. However, the complexity shown by the other metrics demonstrates that it is more understandable and less complex than the original configuration workflow.

Comparing the workflow for the filtered version with both cluster versions, we can conclude that both clusters are more understandable due to lower values related to the four complexity metrics, (i.e., density, *CC*, *CNC* and *CFC*).

In fact, these four metrics help us to know the complexity and understandability of the configuration workflows from the design perspective and the elements in the model. Nevertheless, these metrics used to measure the quality of the workflow are inconclusive to measure the real usefulness and quality of the discovered workflows applied to the context of the variability management.

3.3.2 Analysis of clustering. As introduced in previous sections, two different entropies are applied to infer the clustering (i.e., *Entropy-features* and *Entropy-transitions*). The two entropy formulas can help to understand the quality of the workflow. Thus, a lower value of entropy more quality of the cluster, hence, workflow has more quality. In this regard, Table 4 gives the values regarding the number of clusters and the entropy for each configuration workflow. In this case, the metrics associated with the clusters are aggregated as arithmetic means for better comparison. It is important to highlight that the clusters group a less number of configuration traces, the entropy of the clusters (as mean) are less in both cases than the original and the filtered solution.

Config. Workflow	*N. of Clusters*	*Entropy features*	*Entropy transitions*	*Quality* ($\Delta_{\equiv}$)
Original	1	1	0,021	1598,84
Filtered	1	1	0,027	365,18
Cluster-Features	5	0,188	-	59,02
Cluster-Trans.	3	-	0,0052	115,037

Table 4: Comparison of the number of clusters, entropy and quality.

In order to compare the clusters, the entanglement metric is determined as shown in Figure 9. The entanglement indicates the relation between two dendrograms charts, thus, two different distributions of clusters. The range of the entanglement is [0..1]. The entanglement values closer to 0 are better than to 1. Thus, the entanglement helps to understand how similar are the dendrograms, thereby, how similar clusters are with the independence of the entropy. In this case, the entanglement is 0, 38 which indicates that both clusters are very similar, in other words, the entropy used to perform the clusters reach similar cluster distributions in this case.

As previously mentioned, the uselessness of the quality metrics related to the workflows leads us to define a new custom metric which enables to establish the quality level of the workflow by relating the number of features and their occurrence within the discovered workflow of a cluster. Thus, a metric that enables us

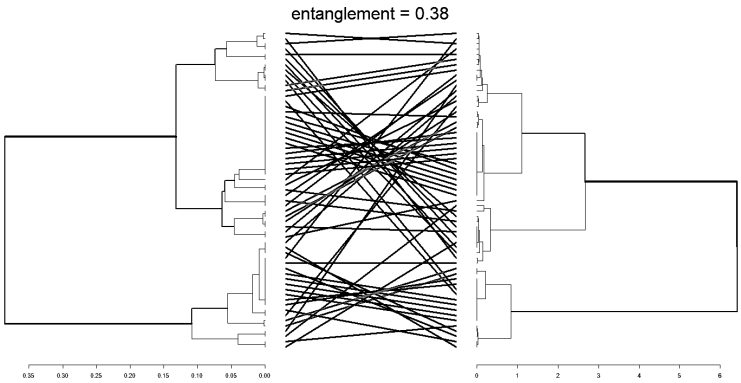

Figure 9: Comparative of entanglement between clusters (*Entropy-features* (right) and *Entropy-transitions* (left)).

to measure how spaghetti is the workflow obtained. Our custom-quality metric is defined as follows:

- Quality ($\Delta_{\equiv}$) measures the difference between the total number of features and the ratio of the sum of the number of times that a feature is selected for each configuration trace and the number of configuration traces.

Formally, given a workflow based on a set of configuration traces (CT) and a set of features ($Features$), the quality can be determined as the following formula:

$$\Delta_{\equiv} = |Features| - \sum_{f \in Workflow} \frac{occurrences(f)}{|CT|} \tag{3}$$

The range of the quality is $[0..|Features|]$, the lower value of quality indicates a better configuration workflow. The number of features and configuration traces are group into the more similar workflow, therefore, it brings about that the quality is near to 0.

For instance, using the example in Figure 3 and the *Cluster 2* in Figure 5, the number included in the rectangle of the feature corresponds to the number of times that the feature is selected regarding the configuration traces. Hence, the quality for the *Cluster 2* can be determined applying the formula as follows:

$$\Delta_{\equiv} = 8 - \left(\frac{3}{3} + \frac{3}{3} + \frac{1}{3} + \frac{1}{3} + \frac{3}{3} + \frac{1}{3} + \frac{2}{3} + \frac{2}{3}\right) \approx 2,67 \tag{4}$$

Comparing the quality results in Table 4, the conclusion is that the original configuration workflow obtains the worst quality and the five clusters obtained using entropy-features have the best quality. Thus, the distribution of configuration traces in the five clusters (cf., Cluster-Features in Figure 8) achieves better results than the original, filtered, even another cluster regarding. In conclusion and according to the defined quality, the Cluster-Features are less complex, more understandable and less spaghetti than the other configuration workflows.

3.4 COLOSSI implementation

COLOSSI is supported by the implementation of a tool which is composed of the next main components:

(1) *Configuration log extractor* is a piece of software module which takes a set of raw configuration log (including timestamps) in a semi-structured format and returns a XES file.
(2) *Configuration log handler* is another piece of software which takes a FM and a XES log as input. First, apply a set of operations over the FM as described in Section 2.3. Then, a data cleaning is carried out over the XES log to get a filtered configuration log. The output of this connector is a new XES log with the filtered configuration log.
(3) *Cluster generator* is a Python/R module which takes a XES log file which is translated into a matrix. This matrix enables the entropy calculation and based on the analysis of certain parameters and the dendrogram, the number of clusters is determined. Using this information, a hierarchical agglomerative clustering algorithm is applied to determine the clusters. A new XES log file is generated for each cluster that composed the final output of this component.
(4) *Discovery connector* is a piece of software which gets the XES file logs of each cluster and automatically feed the ProM to discover the process models by means of the *Inductive Miner*. The output of this component is a process model in Petri-net or BPMN format.

All the resources, thus, configuration logs, the XES files, the workflows discovered, the source code of the COLOSSI tool (i.e., git repository), and a Jupyter notebook that are employed in this work are freely available at [2]http://www.idea.us.es/splc2019/. The notebook is self-explanatory and allows users to work interactively by executing step-by-step instructions to get the clusters.

3.5 Threats to validity

Even though, the experiments presented in this paper provide evidences that the solution proposed is valid, there are some assumptions that we made that may affect their validity. In this section, we discuss the different threats to validity that affect the evaluation.

External validity. The inputs used for the experiments presented in this paper were either realistic or designed to mimic realistic feature models. However, we do not control the development process and it may have errors and not encode all ERP configurations.

The major threats to the external validity are:

[2]DOI:10.5281/zenodo.3236337

- *Population validity*: the ERP feature model that we used may not represent all ERP realistic products. Note that the model was provided after an anonimisation process. Moreover, the timestamps used to derive the traces were relying on the appearance within the input file without an explicit enumeration. To reduce these threats to validity, we chose a single large model that was used in different studies in literature.
- *Ecological validity*: while external validity, in general, is focused on the generalisation of the results to other contexts (e.g., using other models), the ecological validity if focused on possible errors in the experiment materials and tools used. To avoid as much as possible such threats, we relied on previously existing algorithms to perform the process discovery.

Internal validity: concretely, we developed several metrics that reveals different properties of the workflows, however, there might be characteristics of such workflows that are not revealed.

4 RELATED WORK

In this section, we go through the related work of this research.

Configuration workflows. A formal description of configuration workflows is given in [27]. However, a configuration workflow is a bit different from our definition. An activity of the configuration workflow can be mapped to more than just a feature as in our case. However, our approach is complementary because in the handling process we can group different features as well. Furthermore, although formal semantics and automated support for configuration workflows is presented, no automated mechanism is developed to automatically generate configuration workflows from existing configuration logs. In that sense, our approach complements theirs.

Different possible feature orders are defined in [21]. Those orders are used to build web–based configurators hiding the details of the concrete variability model flavours (e.g., OVM, FMs, CVL, etc.). The orders are built from the structure of the variability model. For instance, in the case of FMs, pre–order, pos–order or in–order can be used to determine the feature order in which features are presented to the user. COLOSSI differs from this approach because we use as input configuration logs to automatically derive and cluster configuration workflows. Our approach can be complementary to [21] because different existing workflows could be also measured using process alignment metrics to determine what's the best feature order to be used.

There exist other approaches [67, 68] focused on the field of product configurator design in which configuration workflows has been tackled from the perspective of machine learning.

Application of process mining in different contexts. In order to discover the processes followed by users or systems analysing event logs, process mining has been applied in several scenarios. Depending on the scenario, different are the points of view that could be used to discover a process, such as the activities executed, persons involved, the resources used, the location where the actions occurs, etc. The versatility of process mining techniques has brought about its application to several scenarios [13], being healthcare [38, 49, 52] and IT [39, 47, 55] the most active areas.

The case studies where event logs are produced by human behaviour interactions are specially complex, derived from the free will capacity of the persons that is not always possible to be modelled. This is the context of this paper, where configuration tasks describe the interaction of users with systems. Previous examples in previous scenarios have been developed, such as [2], to analyse how the users interacts with an enterprise resource planning software, or the applicability of software scenarios analysing how the users interact with software to promote improvements about functional specifications or usability aspects [54]. Software development has also provided a complex scenario where process mining can provide mechanism to improve and optimise the known as software process mining [53]. However, configurability issue has not been analysed before with process mining.

High variability in process mining. When there is a high human interaction, as in configuration processes, spaghetti and lasagna processes tend to be obtained. The occurrence of infrequence activities or non-repeated sequence of activities in the analysed log events bring about the necessity to apply frequency-based filtering solutions [12] and other based on the discovery of chaotic set of activities that an be frequent [60].

The infrequence patterns in process discovery are frequently treated as noise [36], being removed from the log traces to discover a process that represent the most frequent behaviour [56]. Different types of filtering can be performed: (i) filtering the events that are not belong to the mainstream behaviour [12, 56]; (ii) integrating the filtering as a part of the discovery [34, 41, 65, 70]; (iii) filtering traces, in an unsupervised [23] or supervised way [11], and; (iv) including a previous steps for clustering the problem, facilitating the discrimination of traces according to different points of view or dividing different types of behaviour [15, 59].

In summary, up to our knowledge, this is the first solution for workflow retrieval in SPL-related contexts. It is also relevant the use of process mining techniques in new domains. This paper aims at promoting sinergies between these two areas of study.

5 CONCLUDING REMARKS & LESSONS LEARNED

In this paper, we have coped with the problem of extracting the actual workflows used by SPL configurators analysing configuration logs. To discover configuration workflows, we decided to rely on process mining techniques. Moreover, we propose to apply clustering to improve the resulting configuration workflows reducing the complexity and improving their understandability. From our research on configuration workflows, we learned the following important lessons:

(1) **Reduce the complexity of the configuration workflows.** We have defined a mechanism based on clustering to divide the configuration logs into smaller configurations groups to facilitate the understanding of the configuration workflows inferred from configuration logs.
(2) **Quality measurement.** We have defined a set of metrics adapted from business process literature, to measure the quality of the obtained clusters and configuration workflows.
(3) **Improving decisions about configurators.** The clustering creation and the analysis provide information to expert users about the features that used to be configured together

or sequential lists of features that could be integrated in a single feature.

In future work, we plan to develop new variability-oriented metrics that can show the impact of the numbers of features within the workflows, trying to incorporate characteristics of the feature models into the clustering and process discovery. Moreover, we would like to apply this technique to more scenarios and datasets to complement the validation of our proposal, including in the analysis other methods to tackle sppagheti processes. Further, we consider interesting to investigate a proper way to obtain the best distribution clusters automatically for a defined numbers of clusters. From our point of view, it is also relevant to propose multiple uses of the resulting workflows to help in different areas such as reverse engineering or SPL testing.

ACKNOWLEDGEMENT

This work has been partially funded by the Ministry of Science and Technology of Spain through ECLIPSE (RTI2018-094283-B-C33), the Junta de Andalucía via the PIRAMIDE and METAMORFOSIS projects, the European Regional Development Fund (ERDF/FEDER), and the MINECO Juan de la Cierva postdoctoral program. The authors would like to thank the Cátedra de Telefónica "Inteligencia en la Red" of the Universidad de Sevilla for its support.

REFERENCES

[1] 2016. IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams. *IEEE Std 1849-2016* (Nov 2016), 1–50. https://doi.org/10.1109/IEEESTD.2016.7740858

[2] Saulius Astromskis, Andrea Janes, and Michael Mairegger. 2015. A Process Mining Approach to Measure How Users Interact with Software: An Industrial Case Study. In *Proceedings of the 2015 International Conference on Software and System Process (ICSSP 2015)*. ACM, New York, NY, USA, 137–141. https://doi.org/10.1145/2785592.2785612

[3] A. Augusto, R. Conforti, M. Dumas, M. L. Rosa, F. M. Maggi, A. Marrella, M. Mecella, and A. Soo. 2019. Automated Discovery of Process Models from Event Logs: Review and Benchmark. *IEEE Transactions on Knowledge and Data Engineering* 31, 4 (April 2019), 686–705. https://doi.org/10.1109/TKDE.2018.2841877

[4] Frank B Baker and Lawrence J Hubert. 1975. Measuring the power of hierarchical cluster analysis. *J. Amer. Statist. Assoc.* 70, 349 (1975), 31–38.

[5] Geoffrey H Ball and David J Hall. 1965. *ISODATA, a novel method of data analysis and pattern classification.* Technical Report. Stanford research inst Menlo Park CA.

[6] David. Benavides, Sergio. Segura, and Antonio. Ruiz-Cortés. 2010. Automated analysis of feature models 20 years later. *Information Systems* 35, 6 (2010), 615–636.

[7] David Benavides, Pablo Trinidad, Antonio Ruiz Cortés, and Sergio Segura. 2013. *FaMa.* Springer Berlin Heidelberg, Chapter FaMa, 163–171. https://doi.org/10.1007/978-3-642-36583-6-11

[8] Jan Bosch. 2018. The Three Layer Product Model: An Alternative View on SPLs and Variability. In *Proceedings of the 12th International Workshop on Variability Modelling of Software-Intensive Systems, VAMOS 2018, Madrid, Spain, February 7-9, 2018.* 1. https://doi.org/10.1145/3168365.3168366

[9] Tadeusz Caliński and Jerzy Harabasz. 1974. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods* 3, 1 (1974), 1–27.

[10] Jorge Cardoso. 2005. Control-flow complexity measurement of processes and Weyuker's properties. In *6th International Enformatika Conference*, Vol. 8. 213–218.

[11] Hsin-Jung Cheng and Akhil Kumar. 2015. Process mining on noisy logs - Can log sanitization help to improve performance? *Decision Support Systems* 79 (2015), 138–149. https://doi.org/10.1016/j.dss.2015.08.003

[12] Raffaele Conforti, Marcello La Rosa, and Arthur H. M. ter Hofstede. 2017. Filtering Out Infrequent Behavior from Business Process Event Logs. *IEEE Trans. Knowl. Data Eng.* 29, 2 (2017), 300–314. https://doi.org/10.1109/TKDE.2016.2614680

[13] Dusanka Dakic, Darko Stefanovic, Ilija Cosic, Teodora Lolic, and Milovan Medojevic. 2018. BUSINESS APPLICATION: A LITERATURE REVIEW. In *29TH DAAAM INTERNATIONAL SYMPOSIUM ON INTELLIGENT MANUFACTURING AND AUTOMATION.* https://doi.org/10.2507/29th.daaam.proceedings.125

[14] David L Davies and Donald W Bouldin. 1979. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence* 2 (1979), 224–227.

[15] Massimiliano de Leoni, Wil M. P. van der Aalst, and Marcus Dees. 2016. A general framework for correlating, predicting and clustering dynamic behavior based on event logs. *Inf. Syst.* 56 (2016), 235–257. https://doi.org/10.1016/j.is.2015.07.003

[16] Richard O Duda, Peter E Hart, et al. 1973. *Pattern classification and scene analysis.* Vol. 3. Wiley New York.

[17] Joseph C Dunn. 1974. Well-separated clusters and optimal fuzzy partitions. *Journal of cybernetics* 4, 1 (1974), 95–104.

[18] A. Durán, D. Benavides, S. Segura, P. Trinidad, and A. Ruiz-Cortés. 2017. FLAME: a formal framework for the automated analysis of software product lines validated by automated specification testing. *SOSYM* 16, 4 (2017), 1049–1082. https://doi.org/10.1007/s10270-015-0503-z

[19] Alexander Felfernig, Lothar Hotz, Claire Bagley, and Juha Tiihonen. 2014. *Knowledge-Based Configuration.*

[20] T Frey and H Van Groenewoud. 1972. A cluster analysis of the D2 matrix of white spruce stands in Saskatchewan based on the maximum-minimum principle. *The Journal of Ecology* (1972), 873–886.

[21] J.A. Galindo, D Dhungana, R Rabiser, D Benavides, G Botterweck, and P. Grünbacher. 2015. Supporting distributed product configuration by integrating heterogeneous variability modeling approaches. *Information and Software Technology* 62, 1 (2015), 78–100.

[22] José A. Galindo, David Benavides, Pablo Trinidad, Antonio-Manuel Gutiérrez-Fernández, and Antonio Ruiz-Cortés. 2018. Automated analysis of feature models: Quo vadis? *Computing* (11 Aug 2018). https://doi.org/10.1007/s00607-018-0646-1

[23] Lucantonio Ghionna, Gianluigi Greco, Antonella Guzzo, and Luigi Pontieri. 2008. Outlier Detection Techniques for Applications. In *Foundations of Intelligent Systems*, Aijun An, Stan Matwin, Zbigniew W. Raś, and Dominik Ślęzak (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 150–159.

[24] Maria Halkidi, Michalis Vazirgiannis, and Yannis Batistakis. 2000. Quality scheme assessment in the clustering process. In *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 265–276.

[25] John A Hartigan. 1975. Clustering algorithms. (1975).

[26] B. F. A. Hompes, J. C. A. M. Buijs, Wil M. P. van der Aalst, P. M. Dixit, and J. Buurman. 2017. Detecting Changes in Process Behavior Using Comparative Case Clustering. In *Data-Driven Process Discovery and Analysis*, Paolo Ceravolo and Stefanie Rinderle-Ma (Eds.). Springer International Publishing, 54–75.

[27] Arnaud Hubaux, Andreas Classen, and Patrick Heymans. 2009. Formal Modelling of Feature Configuration Workflows. In *Proceedings of the 13th International Software Product Line Conference (SPLC '09)*. Carnegie Mellon University, Pittsburgh, PA, USA, 221–230. http://dl.acm.org/citation.cfm?id=1753235.1753266

[28] A Hubaux, P b Heymans, P.-Y Schobbens, D Deridder, and E.K.a Abbasi. 2013. Supporting multiple perspectives in feature-based configuration. *SOSYM* 12, 3 (2013), 641–663. https://doi.org/10.1007/s10270-011-0220-1

[29] Lawrence J Hubert and Joel R Levin. 1976. A general statistical framework for assessing categorical clustering in free recall. *Psychological bulletin* 83, 6 (1976), 1072.

[30] A. K. Jain, M. N. Murty, and P. J. Flynn. 1999. Data Clustering: A Review. *ACM Comput. Surv.* 31, 3 (Sept. 1999), 264–323. https://doi.org/10.1145/331499.331504

[31] Ari Kobren, Nicholas Monath, Akshay Krishnamurthy, and Andrew McCallum. 2017. A Hierarchical Algorithm for Extreme Clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17)*. ACM, New York, NY, USA, 255–264.

[32] Wojtek J Krzanowski and YT Lai. 1988. A criterion for determining the number of groups in a data set using sum-of-squares clustering. *Biometrics* (1988), 23–34.

[33] L Lebart, A Morineau, and M Piron. 2000. Statistique exploratoire multidimensionnelle, Dunod, Paris, France. (2000).

[34] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. 2014. Discovering Block-Structured Process Models from Incomplete Event Logs. In *Petri Nets (Lecture Notes in Computer Science)*, Vol. 8489. Springer, 91–110.

[35] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. 2015. Scalable Process Discovery with Guarantees. In *Enterprise, Business-Process and Information Systems Modeling*, Khaled Gaaloul, Rainer Schmidt, Selmin Nurcan, Sérgio Guerreiro, and Qin Ma (Eds.). Springer International Publishing, Cham, 85–101.

[36] Linh Thao Ly, Conrad Indiono, Jürgen Mangler, and Stefanie Rinderle-Ma. 2012. Data Transformation and Semantic Log Purging for Process Mining. In *CAiSE (Lecture Notes in Computer Science)*, Vol. 7328. Springer, 238–253.

[37] David J. C. MacKay. 2002. *Information Theory, Inference & Learning Algorithms.* Cambridge University Press, New York, NY, USA.

[38] R. S. Mans, M. H. Schonenberg, M. Song, W. M. P. van der Aalst, and P. J. M. Bakker. 2009. Application of Process Mining in Healthcare – A Case Study in a Dutch Hospital. In *Biomedical Engineering Systems and Technologies*, Ana Fred, Joaquim Filipe, and Hugo Gamboa (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 425–438.

[39] Laura Măruşter and Nick R. T. P. van Beest. 2009. Redesigning business processes: a methodology based on simulation and techniques. *Knowledge and Information Systems* 21, 3 (25 Jun 2009), 267. https://doi.org/10.1007/s10115-009-0224-0

[40] Laura Maruster, A. J. M. M. Weijters, Wil M. P. van der Aalst, and Antal van den Bosch. 2002. : Discovering Direct Successors in Process Logs. In *Discovery Science,*

5th International Conference, DS 2002, Lübeck, Germany, November 24-26, 2002, Proceedings. 364–373. https://doi.org/10.1007/3-540-36182-0_37

[41] Laura Maruster, A. J. M. M. Weijters, Wil M. P. van der Aalst, and Antal van den Bosch. 2006. A Rule-Based Approach for Process Discovery: Dealing with Noise and Imbalance in Process Logs. *Data Min. Knowl. Discov.* 13, 1 (2006), 67–87.

[42] John O McClain and Vithala R Rao. 1975. Clustisz: A program to test for the quality of clustering of a set of objects. *JMR, Journal of Marketing Research (pre-1986)* 12, 000004 (1975), 456.

[43] Jan Mendling. 2008. *Metrics for Business Process Models*. Springer Berlin Heidelberg, Berlin, Heidelberg, 103–133. https://doi.org/10.1007/978-3-540-89224-3_4

[44] Glenn W Milligan. 1980. An examination of the effect of six types of error perturbation on fifteen clustering algorithms. *Psychometrika* 45, 3 (1980), 325–342.

[45] Glenn W Milligan. 1981. A monte carlo study of thirty internal criterion measures for cluster analysis. *Psychometrika* 46, 2 (1981), 187–199.

[46] Juliana Alves Pereira, Pawel Matuszyk, Sebastian Krieter, Myra Spiliopoulou, and Gunter Saake. 2018. Personalized recommender systems for product-line configuration processes. *Computer Languages, Systems & Structures* 54 (2018), 451–471. https://doi.org/10.1016/j.cl.2018.01.003

[47] José Miguel Pérez-Álvarez, Alejandro Maté, María Teresa Gómez López, and Juan Trujillo. 2018. Tactical Business-Process-Decision Support based on KPIs Monitoring and Validation. *Computers in Industry* 102 (2018), 23–39.

[48] Ricardo Pérez-Castillo, María Fernéndez-Ropero, and Mario Piattini. 2019. Business process model refactoring applying IBUPROFEN. An industrial evaluation. *Journal of Systems and Software* 147 (2019), 86 – 103. https://doi.org/10.1016/j.jss.2018.10.012

[49] Lua Perimal-Lewis, David Teubner, Paul Hakendorf, and Chris Horwood. 2016. Application of process mining to assess the data quality of routinely collected time-based performance data sourced from electronic health records by validating process conformance. *Health informatics journal* 22 4 (2016), 1017–1029.

[50] DA Ratkowsky and GN Lance. 1978. Criterion for determining the number of groups in a classification. (1978).

[51] F James Rohlf. 1974. Methods of comparing classifications. *Annual Review of Ecology and Systematics* 5, 1 (1974), 101–113.

[52] Anne Rozinat, Ivo S. M. de Jong, Christian W. Günther, and Wil M. P. van der Aalst. 2009. Process Mining Applied to the Test Process of Wafer Scanners in ASML. *IEEE Trans. Systems, Man, and Cybernetics, Part C* 39, 4 (2009), 474–479.

[53] Vladimir Rubin, Christian W. Günther, Wil M. P. van der Aalst, Ekkart Kindler, Boudewijn F. van Dongen, and Wilhelm Schäfer. 2007. Process Mining Framework for Software Processes. In *Software Process Dynamics and Agility*, Qing Wang, Dietmar Pfahl, and David M. Raffo (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 169–181.

[54] Vladimir A. Rubin, Alexey A. Mitsyuk, Irina A. Lomazova, and Wil M. P. van der Aalst. 2014. Process Mining Can Be Applied to Software Too!. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '14)*. ACM, New York, NY, USA, Article 57, 8 pages. https://doi.org/10.1145/2652524.2652583

[55] Mahdi Sahlabadi, Ravie Chandren Muniyandi, and Zarina Shukur. 2014. Detecting abnormal behavior in social network websites by using a process mining technique. *Journal of Computer Science* 10, 3 (2014), 393–402. https://doi.org/10.3844/jcssp.2014.393.402

[56] Mohammadreza Fani Sani, Sebastiaan J. van Zelst, and Wil M. P. van der Aalst. 2017. Improving Process Discovery Results by Filtering Outliers Using Conditional Behavioural Probabilities. In *Business Process Management Workshops - BPM 2017 International Workshops, Barcelona, Spain, September 10-11, 2017, Revised Papers*. 216–229. https://doi.org/10.1007/978-3-319-74030-0_16

[57] Pierre-Yves Schobbens, Patrick Heymans, Jean-Christophe Trigaux, and Yves Bontemps. 2007. Generic semantics of feature diagrams. *Computer Networks* 51, 2 (2007), 456–479. https://doi.org/10.1016/j.comnet.2006.08.008

[58] Steven She, Rafael Lotufo, Thorsten Berger, Andrzej Wasowski, and Krzysztof Czarnecki. 2010. The Variability Model of The Linux Kernel.. In *VAMOS*, Vol. 10. 45–51.

[59] Minseok Song, Christian W. Günther, and Wil M. P. van der Aalst. 2009. Trace Clustering in. In *Business Process Management Workshops*, Danilo Ardagna, Massimo Mecella, and Jian Yang (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 109–120.

[60] Niek Tax, Natalia Sidorova, and Wil M. P. van der Aalst. 2019. Discovering more precise process models from event logs by filtering out chaotic activities. *J. Intell. Inf. Syst.* 52, 1 (2019), 107–139. https://doi.org/10.1007/s10844-018-0507-6

[61] T Thüm, S Apel, C Kästner, I Schaefer, and G.a Saake. 2014. A classification and survey of analysis strategies for software product lines. *ACMCS* 47, 1 (2014). https://doi.org/10.1145/2580950

[62] Wil M. P. van der Aalst. 2011. *Analyzing "Spaghetti Processes"*. Springer Berlin Heidelberg, Berlin, Heidelberg. 301–317 pages. https://doi.org/10.1007/978-3-642-19345-3_12

[63] Wil M. P. van der Aalst. 2016. *Process Mining - Data Science in Action, Second Edition*. Springer.

[64] Boudewijn F. van Dongen, Ana Karla A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and Wil M. P. van der Aalst. 2005. The ProM Framework: A New Era in Process Mining Tool Support. In *Applications and Theory of Petri Nets 2005, 26th International Conference, ICATPN 2005, Miami, USA, June 20-25, 2005, Proceedings*. 444–454. https://doi.org/10.1007/11494744_25

[65] Seppe K. L. M. vanden Broucke and Jochen De Weerdt. 2017. Fodina: A robust and flexible heuristic process discovery technique. *Decision Support Systems* 100 (2017), 109–118. https://doi.org/10.1016/j.dss.2017.04.005

[66] Angel Jesus Varela-Vaca and Rafael M. Gasca. 2013. Towards the automatic and optimal selection of risk treatments for business processes using a constraint programming approach. *Information & Software Technology* 55, 11 (2013), 1948–1973.

[67] Yue Wang and Mitchell Tseng. 2014. Attribute selection for product configurator design based on Gini index. *International Journal of Production Research* 52, 20 (2014), 6136–6145. https://doi.org/10.1080/00207543.2014.917216

[68] Yue Wang and Mitchell M. Tseng. 2011. Adaptive attribute selection for configurator design via Shapley value. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 25, 2 (2011), 185–195. https://doi.org/10.1017/S0890060410000624

[69] Joe H Ward Jr. 1963. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association* 58, 301 (1963), 236–244.

[70] A. J. M. M. Weijters and J. T. S. Ribeiro. 2011. Flexible Heuristics Miner (FHM). In *CIDM*. IEEE, 310–317.