Assessing product quality from the production process health status

Le Toan Duong[1,2,3] and **Louise Travé-Massuyès**[1,2] and **Audine Subias** [1,2] and **Nathalie Barbosa Roa**[2,3]
[1]LAAS-CNRS, Université de Toulouse, CNRS, INSA, Toulouse, France
e-mail: {ltduong,louise,subias}@laas.fr
[2] ANITI, Université Fédérale Toulouse Midi Pyrénées, France
[3] Vitesco Technologies France SAS, Toulouse, France
e-mail: nathalie.barbosa.roa@continental-corporation.com

Abstract

In the manufacturing industry, it is of prime importance to be able to retrieve the health status of production lines and to know how products navigate across operations. Products that are suspected to be faulty are deviated from the nominal path and inspected more closely. The fact that some products deviate from the nominal path and others fail at some check operations might indicate a risk for quality. This paper proposes a method to obtain the process model automatically, following the principles of the process mining field. The found model not only shows the product transitions from one operation to another, but also includes the time constraints bounding the "normal" time for each transition. From this model, products can be clustered according to the path they follow along the process The clustering results for the product batch and the product itself are ultimately used to compute a quality index. The whole method is applied to a SMT PCB production line from *Vitesco Technologies.*

1 Introduction

In the Printed circuit boards (PCB) industry today, parts are manufactured continuously 7 days a week, 24 hours a day. In this paper, we present a quality index for produced parts based on data collected from the assembly processes. To do this, we applied different approaches of process mining to discover and represent the process model as well as to check the conformance of the product production path. Along the path, products are submitted to a series of operations, some of which are control processes that can sanction a product as good or faulty .

The field of process mining is concerned with extracting useful information about process execution, by analyzing event logs [Song *et al.*, 2008]. Process discovery and conformance checking are the most important tasks in the process mining field. The former consists of finding a process model by analysing a set of sequences or traces extracted from event log, while the later refers to the analysis of relation between the behaviour of a process described in a process model and event logs that have been recorded during the execution of the process [Carmona *et al.*, 2018] . Many process discovery algorithms were proposed, among them the α algorithm [Song and van der Aalst, 2008], Heuristic Miner [Weijters and Ribeiro, 2011][Weijters *et al.*, 2006], and Inductive Miner [Leemans *et al.*, 2013]. The algorithm that we propose in this paper used Directly-Follows Graphs (DFGs) for process representation and frequency-based filtering to tackle complexity problem. Conformance checking is useful in several tasks, like performance analysis [Adriansyah and Buijs, 2012], detecting high-level deviations [Adriansyah *et al.*, 2013], and alignment-based precision measures [Adriansyah *et al.*, 2015]. Our conformance checking approach is original in the sense that it uses a decision tree with various criteria to compare real behaviour with the process model taking into account temporal constraints.

Figure 1, taken from [Son *et al.*, 2014], summarizes the process mining framework in a manufacturing process environment. In the data preparation step (step 1), we extracted raw data in form of text messages from MES databases and converted them to event log. Then, the event logs were preprocessed by checking missing values, filtering out redundant data and duplicated values (step 2). In the Process Mining and Analysis step (step 3), for our use case, we focused on process perspective that consists of discovering process model, conformance checking and other performance analysis. Finally, results and evaluation are presented in step 4. These tasks are presented in more detail all along the paper.

In our work, process discovery and conformance checking are ultimately used to sort products in each batch into different classes that can be associated to a penalty index based on the path normal or faulty .These indexes were decided by process experts and used to compute the quality score of product batches. The data that we analyzed comes from different machines and assembly lines producing a single family of products in a real PCB assembly process.

The paper is organized as follows. Section 2 describes our use case, the SMT PCB Assembly process . Section 3 presents some basic mathematical concepts, the process model generation method as well as related time constraints. Section 4 explains how product populations are clustered and the penalty indexes that are used to assess product quality. Section 5 presents the results that we have obtained on our use case. Finally, Section 6 concludes the work and provides some perspective ideas.

This project is supported by ANITI, the "Artificial and Natural Intelligence Toulouse Institute", through the French "Investing for the Future – PIA3" program under the Grant agreement n°ANR-19-PI3A-0004.

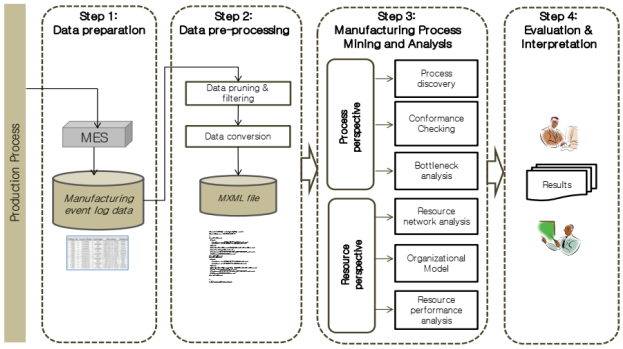


Figure 1: Process mining Framework in a manufacturing environment [Son *et al.*, 2014]

2 Use case: a PCB Assembly process

The PCB assembly process is divided into two main phases (see Figure 2) :

Front End Assembly (FE): electronic components are placed and soldered onto the PCB using Surface Mount Technology (SMT). A schematic view of a SMT line is presented in Figure 2. The first operation is achieved by the Solder Paste Printing (SPP) that prints a solder paste layer on the surface of the PCB. This is followed by the Solder Paste Inspection (SPI) machine. Next, electrical components are mounted onto the surface of the PCB and pass through the reflow oven in order to be permanently attached to the PCB . The front end phase ends with an Automated Optical Inspection (AOI) that checks components position and solders quality. This whole process can be repeated for double-sided boards, in which case we denote **FE1** and **FE2** the two phases.

Back End Assembly (BE): connectors that communicate with peripherals are fixed to the PCB and the set is assembled in a housing for protection and thermal dissipation. Next, functional tests that verify the product operation are finally performed . Back end lines are suited to the specific needs of each product, and in many cases, employ a combination of human operators and specialized robots [Shaw, 2012].

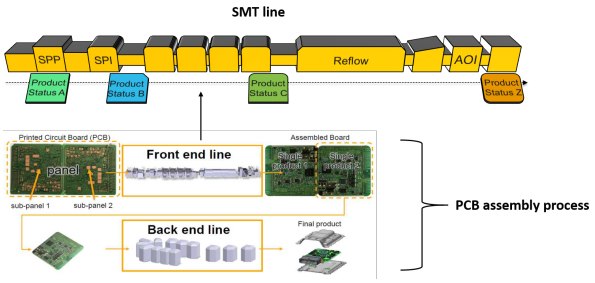


Figure 2: A schematic view of PCB assembly process

3 Timed process model construction

In this section, we introduce several primary concepts used in PM, then the notion of timed process model and how to construct it from event logs. The aim of a timed process model is, on the one hand, to represent the real process and, on the other hand, to represent time constraints that will be used for product's quality characterization.

Let us formally define the concept of *event* which relies on the concept of *event type* expressing the semantic label associated with the event. The set of all event types is denoted by $\mathcal{E}$, $\mathcal{E}^*$ is the set of all finite sequences over $\mathcal{E}$.

Definition 1 (Event)**.** *An event is defined as a pair* (e_i, t_i)*, where* $e_i \in \mathcal{E}$ *is an event type that identifies the event and* t_i *is the event date.*

Time representation relies on the time point algebra and time is considered as a linearly ordered discrete set of instants whose resolution is sufficient to capture the process dynamics. Events generally come in streams forming *sequences*.

Definition 2 (Sequence)**.** *A sequence* $\mathcal{S} \in \mathcal{E}^* \times \mathbb{N}$ *is an ordered set of events denoted* $\mathcal{S} = \langle (e_i, t_i) \rangle_{i \in \mathbb{N}_l} = \{(e_i, t_i) \quad / \quad e_i \in \mathcal{E}, i = 1, \ldots, l, t_i < t_{i+1}, i = 1, \ldots, l-1\}$*, where* l *is the dimension of the sequence* $\mathcal{S}$*.*

Given a set of process activities, events are useful to represent whether the beginning or the end of each activity. Event types provide an identifier for the activity and the date place them in an ordered sequence.

A process instance i.e a product manufacturing is characterized by the sequence of activities that are executed, hence by a sequence of events qualified as *trace*. An *event log* gathers several traces.

Definition 3 (Trace)**.** *A trace is a sequence of events* $\sigma \in \mathcal{E}^* \times \mathbb{N}$ *executed for some process instance. The trace support* $\mathcal{E}_\sigma$ *is given by the ordered set of event types present in the trace.*

Definition 4 (EventLog)**.** *An event log* $\mathcal{L}$ *is a multi-set over* $\mathcal{E}^*$*, i.e., a trace can appear multiple times in an event log.*

Definition 5. *(Directly Follows Graph) A Directly Follows Graph, denoted as* DFG *is a pair* (G, L) *such that:*

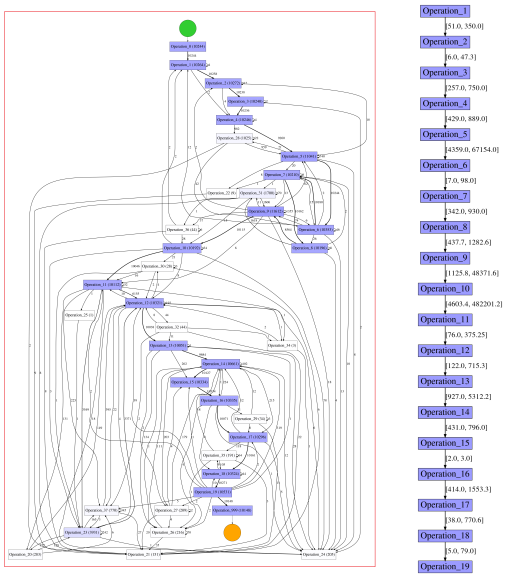
- G *is a directed graph* $G = (V, E)$ *with vertex set* V *and edge set* E*.*
- L *is a set of labels, where* $l_{i,j}$ *is associated with the edge between vertex* v_i *and vertex* v_j*.*

Definition 6. *(Process model) A process model, denoted as* PM *is a* DFG*, where* V *is instantiated with the set of event types related to the process and the edges of* E *represent the precedence relation according to the traces of the process.* V *includes two specific nodes, a source node* V_{start} *and a well node* V_{end}*. The edge labels in* L *can represent different information, such as transition counts, durations, etc.*

In our real use case, V is the set of event types occurring in the traces of our production process, E is the set of all possible transitions representing product state changes, and L contains information about the frequency of these state changes stored in the event log.

Figure 3a shows the DFG of our production process obtained from all event logs data collected in 2019. The graph was constructed with the PM library developed in Python PM4Py [Berti *et al.*, 2019]. In this graph, the event types associated to nodes are numeric identifiers. The edges refer to product state transitions, for example between operations. Additionally, the label on top of each edge, $l_{i,j}$ represents the number of products that have taken the corresponding transition. For example, there are 10244 product transitions between the node marked with the event type Operation 0

(the *Laser Printing* operation) and the node marked Operation 1 that corresponds to the *Solder Paste Inspection face 1*.



(a)

(b)

Figure 3: Process model (a) Directly-Follows Graph. (b) Timed process model.

The constructed PM is a representation of all the transitions that have been undertaken by the products in 2019.

In the first instance, we used a simple algorithm to find the nominal process model. Before describing this, let us notice that a trace as defined in definition 3 corresponds to a path between V_{start} and V_{end} in PM. The PM shown in Figure 3a includes various such paths. In our algorithm, the nominal process model is constructed based on the path that most products follow. We call this path the *nominal path* $\mathcal{P}^*$. Products of the same family follow the same path if nothing wrong happens along the production chain. The nominal path $\mathcal{P}^*$ is computed by the following formula:

$$\mathcal{P}^* = \underset{\mathcal{P} \in PM}{\operatorname{argmax}}\, freq(\mathcal{P}) \tag{1}$$

where $freq(\mathcal{P})$ returns the number of traces, i.e. process instances, that follow the path $\mathcal{P}$.

In Figure 3, the nominal path can be retrieved by following purple color operation identifiers.

3.1 Time constraint extraction

Time constraints are quite indicative of process problems and this is why we are interested in retrieving what would be a normal time between operations. To do so, we opted for an interval time label that represents the most frequent time support among products.

Let us consider an event log $\mathcal{L}$ that gathers a set of traces $\mathcal{T}$ representing different instances of the same process PM. Considering two adjacent edges v_i and v_j corresponding to two event types e_i and e_j belonging to the support of all the traces in the subset $\mathcal{T}_k \subseteq \mathcal{T}$, we assume that in any trace $\sigma \in \mathcal{T}_k$, $t_j > t_i$, then the label l_{ij} is determined as follows:

$$l_{ij} = [t^-_{i,j}, t^+_{i,j}] \tag{2}$$

where:

$$t^-_{i,j} = min_{e_i,e_j \in \mathcal{E}_\sigma, \sigma \in \mathcal{T}_k}(t_j - t_i),$$
$$t^+_{i,j} = max_{e_i,e_j \in \mathcal{E}_\sigma, \sigma \in \mathcal{T}_k}(t_j - t_i).$$

The time intervals defined in equation (2) represent the min and max bounds on the elapsed time between two consecutive product state changes in the process.

3.2 Timed process model

In the previous paragraph, we presented the time constraint extraction method. The process model labelled with these time intervals is called the *Timed process model*. Its definition is given as follows:

Definition 7. ***(Timed process model)*** *A timed process model (t-PM) is a process model PM for which edges are labelled according to the time constraints given by equation (2).*

A process instance, or a trace $\sigma = \langle (e_1, t_1), (e_2, t_2), ..., (e_n, t_n) \rangle$ is said to satisfy the timed process model if:

1. The sequence of event types $\langle e_1, e_2, ..., e_n \rangle$ can be replayed in the graph G of t-PM.
2. All event pairs (e_i, t_i) and (e_j, t_j) satisfy the time constraint $[t^-_{i,j}, t^+_{i,j}]$, i.e. $t_j - t_i \in [t^-_{i,j}, t^+_{i,j}]$.

4 Characterizing production quality

The timed process model obtained in 3.2 provides a real model of the process but it is also a support to better characterize the production quality i.e the quality of final products. The idea is to sort the product populations and to associate each population with a penalty index depending on the faulty operations that are encountered along the path. For this, we developed a classification algorithm to cluster products into different populations or classes based on their production path (i.e process instance) .

4.1 Sorting populations by conformance checking

Populations are defined by constructing a decision tree for the classification [Myles *et al.*, 2004]. The primary advantage of using a decision tree is that it is easy to follow and understand. Decision trees have four main parts: a root node, internal nodes, leaf nodes and branches. The root node is the starting point of the tree, and both root and internal nodes contain a test or criteria on an attribute. Each branch represents the answer to the test, and each leaf node represents a class label. Let $\mathbf{Y} = \{y_1, y_2, ..., y_m\}$ be a set of m class labels for all process instances. The aim of the partition issued from the decision tree is to gather homogeneous process instances together. Classes are then associated with penalty indexes that characterize their property.

For a first study, classes and penalty indexes are defined by process experts. The partition of products into these classes is done by checking the conformance of the corresponding trace with the timed process model. Several criteria are used as the consistency with the time constraints, the order of operations, the presence or absence of some important operations . A description of these criteria or classification rules are presented by a decision tree given in Figure 4. The light blue boxes represent internal nodes that contain

split rules. The gray boxes are leaf nodes with corresponding class labels. Each class is associated with a score or penalty index (red number).

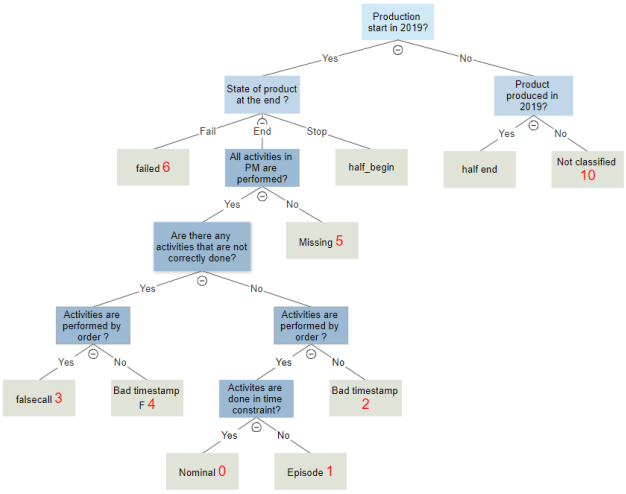


Figure 4: Decision Tree for product classification.

4.2 Penalty index of products

As mentioned previously, each product is associated with a so-called *individual penalty index* depending on which class it belongs to. In our context, this index is not sufficient as we assume that the quality of a product depends mainly on its production path and on its production *batch*. A *batch* is defined as a sub-set of products that are produced continuously and consecutively. Products within a batch are assembled under almost the same conditions and configuration. Then we can extract batches from the event logs by analysing the regularity of products in the production line, i.e. by identifying inactive period. An inactive period is a pre-defined time intervals in which no product passes due to configuration changes in production.

We formally define batches as follows. Let $\mathbf{B} = \{b_j, j = 1, ..., |B|\}$ be a set of $|B|$ batches, where $b_j = \{(x_i^j, y_i^j), i = 1, ..., |b_j|\}$, x_i^j is the i-th product in the batch j, $y_i^j \in \mathbf{Y}$ is the associated class label and $|b_j|$ the batch size, i.e. the number of products that the batch contains. We denote $\tilde{p}(x_i^j)$, the individual penalty index of product x_i^j. For example, a product x_i^j has penalty $\tilde{p}(x_i^j) = 0$ if it belongs to class *nominal* (see Figure 4). This means that this product follows the nominal path without fail operations and respects the temporal constraints between operations.

The *penalty index* of a product in a batch, denoted $p(x_i^j)$, is defined as a combination of its individual penalty index $\tilde{p}(x_i^j)$ and the penalty index of its batch $p(b_j)$.

- **Penalty index of the batch** b_j:

$$p(b_j) = \frac{1}{|b_j|} \sum_{i=1}^{|b_j|} \tilde{p}(x_i^j) \tag{3}$$

- **Penalty index of i-th product in batch** b_j:

$$p(x_i^j) = \lambda \times \tilde{p}(x_i^j) + (1-\lambda) \times p(b_j), \qquad \lambda \in [0, 1] \tag{4}$$

Generally products go through several phases from separate components to assembly and packaging. Due to the heterogeneous in configuration of different phases, the partition of products into batches in each phase is also different. Hence, we compute the penalty index of product after each phase and the penalty index for final product at the end of process is defined as the weighted average of them. Let $\mathbf{X} = \{x_i, i = 1, ..., n\}$ the set of all products. In each phase, a product x_i belongs to a batch b_j and the associated penalty index is computed by formula (4). Let $\mathbf{I}_i$ the set of indexes j such that b_j contains x_i over phases:$\mathbf{I}_i = \{j \in \mathbb{N} | x_i \in b_j\}$. The final penalty index of product x_i noted $p_f(x_i)$ is defined as:

- **Final penalty index of product** x_i:

$$p_f(x_i) = \sum_{j \in \mathbf{I}_i} \alpha_j \times p(x_i^j) \quad , \sum_{j \in \mathbf{I}_i} \alpha_j = 1 \tag{5}$$

Next section presents the application of our proposal.

5 Quality evaluation of the PCB production Line

The proposed approach has been illustrated using data set from a *Vitesco Technologies* (VT) plant, an automotive company specialized in drivetrain technology for all types of vehicles. At VT plants, all products are tracked by an unique ID in form of a data matrix printed on the PCB. This matrix is read each time the PCB goes through an operation. Furthermore, the manufacturing processes are tracked, traced and controlled during the production by the Manufacturing Execution Systems (MES).

5.1 Data description

Raw data used in this use case are generated in real-time or near-real-time by the MES. In fact, machines generate data in form of messages that contain information about production process. Figure 5 presents an example of decoded messages from *Vitesco Technologies* cloud storage service. Each message is generated from every single operation through which a product (PCB) is performed. Hence, messages contain features related to machine, operation and product. We introduce below a list of common features (Figure 5):

- **Type of message (red):** There are mainly two types of message, informative message and control message. While informative messages notify that the PCB has entered or exited some operations, control messages give us detailed information about the quality of product. Hence, these messages have a feature of sanction that could be Pass (P) or Fail (F). An example of control messages is the one generated from the AOI machine.
- **Machine host name/Machine_ID (pink):** The identification of the machine or computer that performed an operation in the PCB.
- **Optional description (violet):** comments about the performed operation.
- **Family (green):** Product family that is being produced.
- **Serial Number/Board_ID (orange):** The identification number of the product.

- **Operation Code/Operation_ID (blue):** The identification number of the operation being performed.
- **Sanction (brown):** (For control message) The sanction given by the operation (F/P), the "F" means the operation has failed meanwhile the "P" letter means the operation has been performed successfully.
- **Timestamp (purple):** The date and time an operation was performed.

```
PS|mid00003|Ope4_Line3|FamilyC128|Board_544|1548866935|
P2|mid00002|Ope7_Line2|FamilyC128|Board_560| Operation_2|FACE2||006380|U55H4583|Line2-Config|2|1548866902|
SF|mid00000|Ope2_Line0|FamilyC128|Board_566| Operation_11|002|P|1548858285|||||||||||||||||||||||||||||
SS|mid00000|Ope15_Line0|FamilyC128|Board_730|Operation_4|P|1546860302|
```

Figure 5: Snapshot of messages file from MES. The data was encoded for preserving confidentiality.

The aim of our study is to analyse the path of products and their transition time through event logs. Hence, we considered only features that allow us to transform messages into event logs. These are *Type of message*, *Serial Number*, *Operation Code*, *Sanction* and *Timestamp*. The experimentation was carried out on data related to a specific product family collected during 2019.
A pre-processing stage was needed in order to removing duplicated messages, filtering bad format messages, gathering and sorting messages related to the same product and transform data into event logs.

The final data set corresponds to over 10000 single electronic boards. We analyzed that over $98,5\%$ (9902 products) of them are good products i.e successfully produced and delivered to clients.

During production process, events tagged with a *fail* notification indicate that the corresponding operation has not been performed as supposed. These fail sanctions can come from a real defect or an equipment measurement error. Diagnosis need to be perform on these products in order to reveal the real defect cause. Figure 6 represents the distribution per product of the number of fail events appearing during production process for *good* and *bad* products. There are more fail events generated in bad products than in good products, which allows us to take into account this in our product quality characterizing approach.

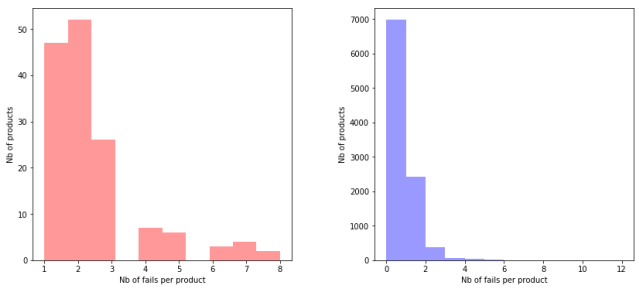


Figure 6: Distribution of Fail events per product. Left: Bad products, Right: Good products.

Moreover, the data analysis pointed out that most products were manufactured within 2 days, as expected, but some of them were in production process longer than that. These products require a more thorough analysis.

5.2 Quality evaluation of electronic boards

For characterizing product quality, the penalty index score from a timed process model as presented in Sections 3.2 and 4 was used.

In this use case, for the time constraint extraction (see Section 3.1) instead of using the min and max value to define the bounds of time constraints, we used a statistical value (5^{th} and 95^{th} percentile) to exclude extreme and outlier values. The process model for this product family is quite simple (see Figure 3b). Operations are performed successively, one after the other. There is no deviation and there are not operations that perform at the same time (problem of concurrency).The next step is to classify products by comparing their production path with the timed process model. Split rules based on the conformance of product's path with the timed process model and classes are presented in the decision tree in Figure 4. With an process expert we also associated to each class a penalty index between 0 an 10 which characterizes the conformance level.

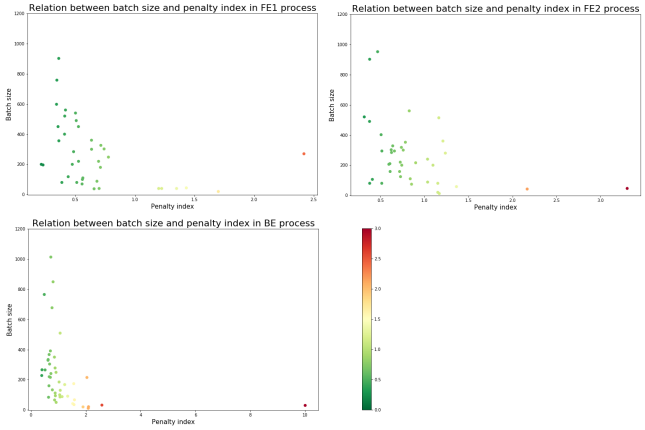


Figure 7: Batch penalty indexes and batch sizes for each production phase

Note that we excluded two classes *half begin* and *half end* because paths in these classes are incomplete.
Once products are classified, we computed penalty index for batches and for final products based on equations (3) and (5).

As a reminder, a batch is a sub-set of products that are produced continuously and consecutively. Batches are extracted from event log and the result shows that the number of products in batch is varied. Figure 7 presents the relation between penalty index and batch size in the three phases $FE1$, $FE2$ and BE of production process. Most of batches have a small penalty index, between 0 and 1 over three phases. This result shows that the majority of batches respect the timed process model. It also shows that there are no perfect batch which has a penalty index of 0. This means that there is no batch in which all products pass through the production line without any failure or deviation and within the accepted time interval. Additionally, Figure 7 shows that batches with a high penalty index (orange points) have small size. Without an in-depth investigation, this correlation is explainable. Indeed, the fact that these batches are small could be due to process interruption or changes in configuration after a sequence of products with bad behaviour. Hence, the penalty index for them is pretty high.

Figure 8 shows the distribution of penalty indexes for products in three phases of production process. As expected, most products have a penalty index in $[0, 1]$ which indicates a good compliance with process model. Moreover, we can see that penalty indexes of products in $FE1$ phase have less variation than in $FE2$ and BE phase. These results open the door for further examination and improvement actions.

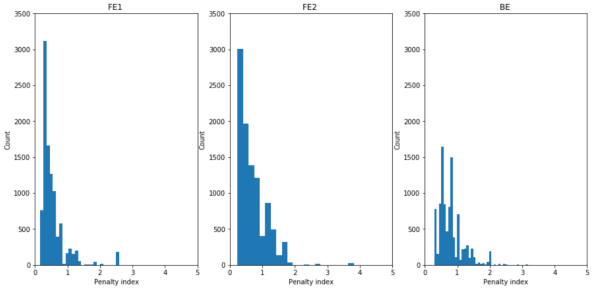


Figure 8: Product penalty indexes for each production phase

Finally, we present below the final penalty index for products as defined in equation (5) (see Figure 9). as expected from previous results most of products have small penalty index. In association with process experts detailed analysis should be performed on products with high penalty index. The results obtained give us relevant information about products behaviours during assembly process. These results were sent to production process experts for further analysis and process optimisation.

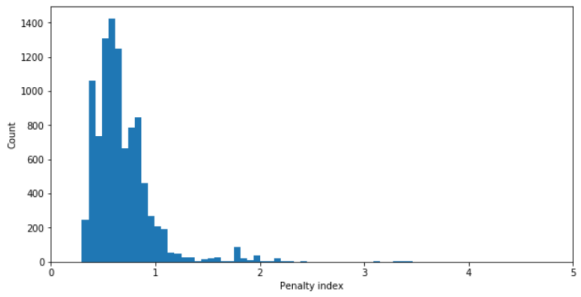


Figure 9: Final penalty indexes for all products.

6 Conclusion and Future work

In this paper, we presented a log-based framework for health status characterization of a production process based on a quality index. Process mining techniques are used to build a timed process model from event logs and to check the conformance of product's paths. An experiment was conducted on a large dataset from the automotive manufacturing company Vitesco Technologies.

The experiment demonstrated that the proposed framework works well. The results showed to be effective for domain experts to have a deeper knowledge of real production process. The proposed indicator evaluates the behaviour of products during their production time. This indicator provides an overview of products circulation and reveals abnormal ones. For future work, we need to fine-tune the model parameters under the supervision of process experts. The products behaviour information found does not seems to be enough to asses their quality. We expect, for the next study, to have information on product quality that we could integrate into our approach to build a learning model for classification and optimizing parameters. The proposed framework is generic and could be applied for various problems in process monitoring and optimisation.

References

[Adriansyah and Buijs, 2012] Arya Adriansyah and Joos CAM Buijs. Mining process performance from event logs. In *International Conference on Business Process Management*, pages 217–218. Springer, 2012.

[Adriansyah *et al.*, 2013] Arya Adriansyah, Boudewijn F Van Dongen, and Nicola Zannone. Controlling break-the-glass through alignment. In *IEEE International Conference on Social Computing*, pages 606–611, 2013.

[Adriansyah *et al.*, 2015] Arya Adriansyah, Jorge Munoz-Gama, Josep Carmona, Boudewijn F van Dongen, and Wil MP van der Aalst. Measuring precision of modeled behavior. *Information systems and e-Business Management*, 13(1):37–67, 2015.

[Berti *et al.*, 2019] Alessandro Berti, Sebastiaan J. van Zelst, and Wil M. P. van der Aalst. Process mining for python (pm4py): Bridging the gap between process- and data science. *ArXiv*, abs/1905.06169, 2019.

[Carmona *et al.*, 2018] Josep Carmona, Boudewijn van Dongen, Andreas Solti, and Matthias Weidlich. *Conformance Checking*. Springer, 2018.

[Leemans *et al.*, 2013] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. Discovering block-structured process models from event logs - a constructive approach. In *Application and Theory of Petri Nets and Concurrency*, pages 311–329. Springer, 2013.

[Myles *et al.*, 2004] Anthony J Myles, Robert N Feudale, Yang Liu, Nathaniel A Woody, and Steven D Brown. An introduction to decision tree modeling. *Journal of Chemometrics*, 18(6):275–285, 2004.

[Shaw, 2012] M.J. Shaw. *Information-Based Manufacturing: Technology, Strategy and Industrial Applications*. Springer US, 2012.

[Son *et al.*, 2014] S Son, Bernardo Nurgroho Yahya, Minseok Song, Sangsu Choi, Jeongho Hyeon, Bumgee Lee, Yong Jang, and Nakyun Sung. Process mining for manufacturing process analysis: a case study. In *2nd Asia Pacific Conference on Business Process Management, Brisbane, Australia*, 2014.

[Song and van der Aalst, 2008] Minseok Song and Wil M. P. van der Aalst. Towards comprehensive support for organizational mining. *Decis. Support Syst.*, 46(1):300–317, December 2008.

[Song *et al.*, 2008] Minseok Song, Christian W Günther, and Wil MP Van der Aalst. Trace clustering in process mining. In *International conference on business process management*, pages 109–120. Springer, 2008.

[Weijters and Ribeiro, 2011] A. Weijters and Joel Ribeiro. Flexible heuristics miner (fhm). *Journal of Applied Physiology*, pages 310–317, 04 2011.

[Weijters *et al.*, 2006] A. Weijters, Wil Aalst, and Alves Medeiros. Process mining with the heuristics miner-algorithm. *Cirp Annals-manufacturing Technology*, 166, 01 2006.