Process Mining in a Manufacturing Company for Predictions and Planning

Milan Pospíšil
Department of Information Systems
BUT, Faculty of Information Technology
Brno, Czech Republic
ipospisil@fit.vutbr.cz

Vojtěch Mates
Department of Information Systems
BUT, Faculty of Information Technology
Brno, Czech Republic
imates@fit.vutbr.cz

Tomáš Hruška
Department of Information Systems
BUT, Faculty of Information Technology
IT4Innovations - Center of excellence
Brno, Czech Republic
hruska@fit.vutbr.cz

Vladimír Bartík
Department of Information Systems
BUT, Faculty of Information Technology
Brno, Czech Republic
bartik@fit.vutbr.cz

***Abstract*—Simulation can be used for analysis, prediction and optimization of business processes. Nevertheless, process models often differ from reality. Data mining techniques can be used to improve these models based on observations of a process and resource behavior from detailed event logs. More accurate process models can be used not only for analysis and optimization, but also for prediction and recommendation as well. This paper analyses process models in a manufacturing company and its historical performance data. Based on the observation, a simulation model can be created and used for analysis, prediction, planning and for dynamic optimization. Focus of this paper is in different data mining problems that cannot be solved easily by well-known approaches like Regression Tree.**

Keywords - business process simulation, business process intelligence, data mining, process mining, prediction, optimization, recommendation, association rules, genetic algorithms.

I. INTRODUCTION

Classic simulation can be used for the analysis of business processes. It is useful to test many variations of processes, measure the effects and then choose the optimal process settings. Thus, the process can be redesigned. It is possible to change resource allocation and search for the most optimal configuration with respect to context-based requirements (price, effectiveness, customer satisfaction, etc.). The current process configuration can be tested to discover how many cases it can handle over periods of time.

These models can be built manually but this is time consuming and error prone. The main disadvantage is that this approach cannot be used for predictions of operational decision, but only for strategic decisions if there exist some dependency on case attributes (see later). The operational decisions are important for internal logistics purposes. The casual models have some simplifications – for example overall probabilities of routing and naive execution time of the task. These parameters are set with respect to on long observation of processes, so they can work in a long-term simulation for strategic decisions. Nevertheless, operational decisions require short-term simulation. These two simulation approaches differ significantly. The short-term simulation starts in the current state of the process with allocated resources, cases in progress with known parameters and with waiting cases to handle. Routing probabilities and execution times can differ significantly for different case variables, therefore, mining of deeper dependencies is needed for better solution.

For example, let us assume a repair process. There are two tasks – repair of a basic item and repair of an advanced item, repair of a basic item is executed in 90% of cases and repair of an advanced item only in 10% of cases. Execution time for a basic item is about one hour and execution time for an advanced item is about eight hours. If our current case has attributes and these attributes lead to advanced repair with 80% probability, classic approach using overall routing probabilities is not precise enough to be used. And there is one another problem – execution time of a task is also influenced by case attributes – some case attributes may lead to longer execution time. Resources have to be also taken into account, e.g., some people work faster and some of them slower.

Predictions, recommendations, and dynamic optimizations could be accomplished by operational simulation. The system can warn us that some cases will be probably late, based on comparison with historic performance data. Then, some different scenarios can be simulated and evaluated. After that, the system can recommend us actions and provide dynamic optimization of current running cases – for example; assigning extra resources from a non-critical case to critical or use a different sub-process – if we have a slower and cheaper or faster but more expensive variation.

This paper analyses processes of a manufacturing company. Simulation model is built using process mining and it is used for predictions. Based on these predictions, managers can change priorities (reallocation of resources) or better plan their storage space, because working front is known and, therefore, they can better predict manufacturing time. Building of simulation model from discovered model is beyond that paper, because analysis of data is also quite hard problem because of many real issues that need to be solved – that leads to some different problems. So this paper is mainly about analysis of data and then, these results could be used for the simulation and for other problems as well.

This work is an extension of our previous research in process mining and simulation field [1]. Paper is organized as follows. Section II describes related papers. Section III is overview of whole idea with some topics as utilization in real processes or problems with data preparation. Our real manufactory is described in Section IV. Then, three different approaches are described in Sections V, VI and VII. It is because different types of real problems that are needed to achieve our goal. Section V describes prediction using standard classification (like regression trees). Section VI solves the same problem but with variable feature vector length. This type of problem could be solved by classification methods, but many of them has problem with variable vectors. So we chose Association Rules to solve this. Both sections are based on existing methods with some extensions to better fit our problem. These extensions are described quite in general, not only for our company. Section VII solves another type of common problems – unmeasured processes. These are processes that are not fully measured and we propose some solutions to deal with it. Also, some further research is consulted. Section VIII is about analyzing errors. Previous sections are about predicting execution times, but errors are also important. Section IX is our summary of experience in real manufacture and about problem with measurement, because it is costly. Last section is conclusion. Experiments are provided in all of three main sections (V, VI, VII and partly VIII) for better intelligibility.

II. RELATED WORK

Data mining techniques can be used in Business Process Management. The research area is referred to as Process Mining [2][3][4][5][6]. It is focused on analysis of information from event logs that were produced by business processes. Process discovery (Figure 1) is one of the methods and it is able to find a process model from an unknown process using many sequence examples of tasks and case parameters.

Except process model, also decision rules, social networks [3][7][8] and simulation models [7][8][9][10] could be discovered.

Resource behaviour is also a point of interest [11][12][13]. Example of simulation model [7] is depicted in Figure 2.

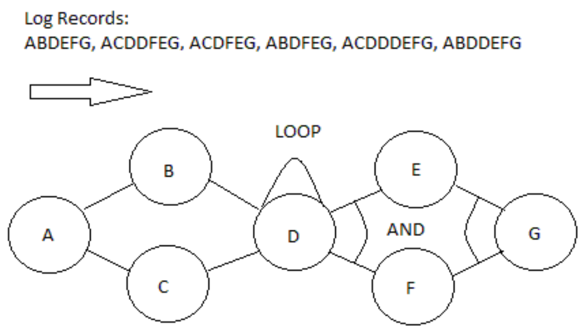

Figure 1. Process discovery. It is possible to discover a process model from the trace log.

It is possible to see routing probabilities and decision rules (decision rules are used when case attributes are known – that leads to better routing rules) and it is possible to see time distribution of tasks.

Some other research on process prediction was published in [8][14][15][16]. Wetzstein [15] used decision trees to analyse process performance (see Figure 3). As it can be seen, if the response time of a banking service is higher than 210, KPI (key performance indicator) is always violated. If customer id is 1234, manager can observe process bottlenecks and try to make banking service faster or find out why the customer 1234 has problems.

Grigori [16][17] uses similar approach used not for analysis but for predictions. Huge classifier is learned based on case attributes, start time and end time of task execution. Classifier can predict final execution time of a case based on case parameters and time information from executed tasks. Evaluation of that approach compared to our approach is discussed in [10]. In addition, our work uses similar approach as [15][16][17] but it combines it with process mining.

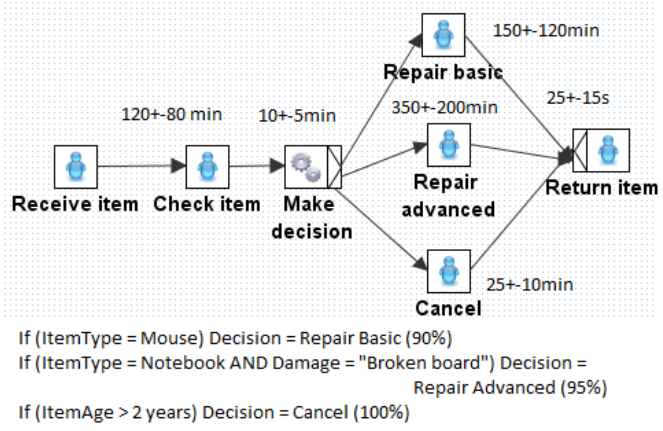
150+-120min
Repair basic
120+-80 min
10+-5min
350+-200min
25+-15s
Receive item
Check item
Make decision
Repair advanced
Return item
25+-10min
Cancel
If (ItemType = Mouse) Decision = Repair Basic (90%)
If (ItemType = Notebook AND Damage = "Broken board") Decision = Repair Advanced (95%)
If (ItemAge > 2 years) Decision = Cancel (100%)

Figure 2. Simulation model. Classic simulation model is enhanced by decision rules. Decision rules can make our routing probabilities more precise, because they depend on case attributes.

Finally, when we discover deeper dependencies between routing rules and execution time of cases, we can use it for simulation related to decision support [10].

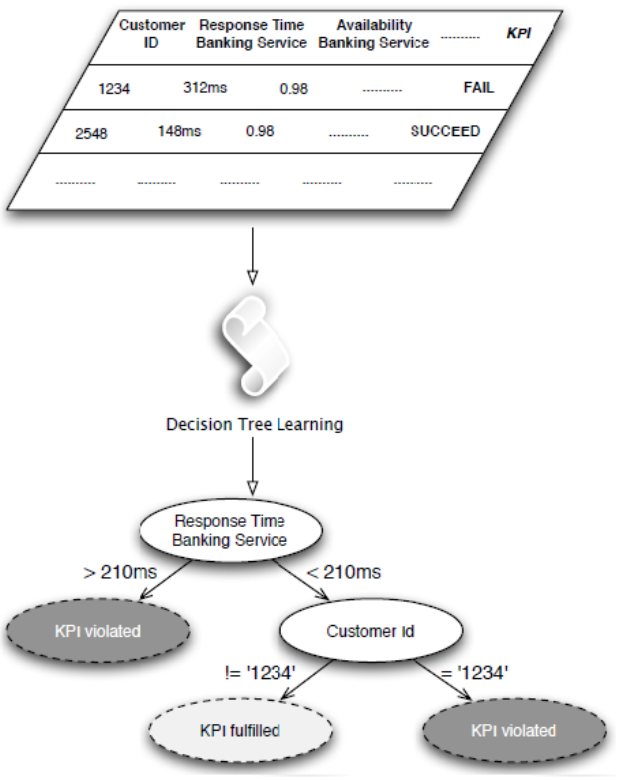

Figure 3. Process performance analysis (taken from [4]). Decision tree is used to discover factors that lead to KPI violation. We can see that KPI is violated when response time of a banking service is larger than 210.

Our previous work [1] is an extension of papers [7][8][10] and it adds some important features, some of them inspired by papers [15][16][17]. For example, execution time of cases could be also predicted by a classifier such as decision rules. This paper shows that our theory of [10] can be applied in a real large manufactory company. It also adds some additional theory in Section III.

III. More Precise Simulation Model

As it is said in [10], there is a demand on building more precise models than the one described by papers [7][8]. We will describe steps needed to accomplish it.

A. Process Discovery

If a process is not known, it is possible to discover it using process discovery techniques [4][6]. However, only process discovery is not sufficient to build simulation model. If an explicit model is not present, it is possible to discover it, but the precision of the model will be lower than the explicitly given by a real model. In some companies, discovered model could be more precise than an official model but it is because these companies do not have their models well-structured in many cases. This is not the case for manufacturing companies where prediction and usage of short time simulation is considered to be better.

B. Decision Mining

Decision mining is based on discovering routing rules in OR split nodes. These rules could be also available but sometimes they are not applicable. Let us assume that a routing rule is based on one parameter, which is inserted into the system just before the decision. Thus, our predictor will know the next path only in the time of decision – this is a useless prediction. In these situations, decision mining has to be used. The topic of decision mining is described in [7], [8][10]. Classifier is learned on training data where inputs are case attributes and output is the next path in process. Our work [10] describes another problem, which is missing attributes or 100% precise attribute known in the time of decision inserted by human (described earlier). If some attributes are missing, classic classifiers will not work in a proper way. If there is a 100% precise attribute then classifier is based only on that attribute. Solution is the same for both problems – it is necessary to build several classifiers for several milestones of the process – from the start (only subset of case attributes are known) to the end (all attributes are known).

C. Execution Time of Tasks

Execution time of tasks is the most important issue in the short-term simulation. Process model and routing rules are important as well. However, in companies with predictable business processes (especially manufacturing companies), control flow and routing rules are used to be formalized and nearly 100% precise.

Execution time of tasks will be described precisely in Sections V, VI and VII where different approaches for different problems will be introduced.

D. Usage of analysis and simulation

There are several ways to use our methods in practice. We must realize that we must do two tasks. First task is the analysis of historic data, which is the main focus of the paper. Second task is to build the simulation model for predictions based on previous data discovery. Second task is not the main topic of this paper, but it is the main goal. Now, we will describe utilization of both steps.

1) Prediction, Planning, Reccomendation

First usage is obvious and we described it in Section I. If we know the task execution times (does not matter if it is the result of data mining or manual measurement), we can better plan our whole work, material flow, inventory, machine and resource utilization. Using simulation, we can evaluate multiple plans and choose the most suitable of them. Also, we can monitor running process and check if some problem is going to come in future. Of course, with some probability – there must be multiple simulation runs.

There is other utilization in scenario testing. Managers can make some change in the process (better machine) to test what is the influence and cost of the change.

2) Analysis

Because the first step is about data mining, we can use these discovered information for another use. If we know what case attribute causes what time and variation, manager could ask for most influential attribute combinations (low or high time and variation). Variation is also very important because high variation suggest some production problems, which is not good for planning. With these tools, managers can focus on most problematic attribute combinations and try to solve them.

Another usage is for worker performance [13]. There is a problem with computation of worker performance, because it is dependent on attributes. Paper [13] describes it in further detail.

Error analysis is also an important part. Some attributes have higher error probability. Errors are important not only to predict execution time of tasks but also to predict routing between tasks – faulty product could be sent to repair and could not almost affect time of task where it occurred. Analysis of error is important for execution time of prediction and there is whole short section at the end of the paper that describes it more deeply.

Last usage is about analysis of changes. Suppose management of company bought new machine (or new working method). Was it really an improvement and how much? If we analyze both old and new task data, we can discover differences between them. For example, new machine could be overall better, but only for some attribute combination.

Analysis of time and variations based on attributes could be also good for advertising. Management could focus on products that have good attribute combinations for manufactory – in our door company, some doors are produced fast, some more slowly. If there is a big commercial on some more problematic doors, production should be late and it could harm name of the company.

3) *Simulation based on historic data*

This type of simulation is not for prediction, but for analysis. It is not the simulation as we know it, but only replaying of historic log data. Using this, we could analyse effect of changes on queues and much more interesting features. But this type of analysis is beyond the scope of our paper.

E. Problems in preprocessing step

Every data mining needs data cleaning and preprocessing. This is the step most dependent on data type. We should describe some problems we encounter in our practice.

1) *Analysing quality of data set*

First step we must always do is to gain information about data set reliability. We must analyse records with the same attributes, if they are available – they may not, but almost every data set contains some set of records with identical attributes. Then, we must compute variance of those similar records. For example, if there are twenty records with attributes A, B, C, we have to compute their variance execution times. If A, B, C = 250s, A, B, C = 300s, A, B, C = 350s, etc., then we have to compute variance from these numbers. If the result variance of nearly all of types of record set is not lower than the variance of whole data set (variance of all record times), the data set is useless. It means there is no dependency on attributes or some attributes are missing or there was some problem in measurement – last problem is the most probable in our experience.

2) *Low and High Values*

Beware extreme values in data sets, it means different set of scenarios. Extremely low values are probably errors that were discovered and sent to repair. Extremely high values are errors or poorly measured values. We must be careful, because sometimes break can affect time dramatically – suppose start of task was measured correctly, but then break occurred and then work continued – total task time will not be useful at all. So high values do not mean automatically error, if we want to analyze errors (see later), we need information about errors explicitly given in data. Another possibility is schedule of breaks.

3) *"Half Measurement"*

Another problem, sometimes solvable, is the "half measurement". Sometimes, there is only information about start or end of the process. Sure, we can deduct these times to get real time. Deduct means that if product A enters into a task and (for example) start time is measured, then product B enters the machine and another start time is measured. Then, if a task could contain only one product, we can compute Time as Start(B) – Start(A). It is easy, but it may not work. If we are deducting times (start or end), there must be quite high utilization of task (low waiting for product processing – pause). Every waiting could be considered as production because we have no idea if there was waiting or our product took more time to produce.

Fortunately, this problem could be sometimes solved, but only in tasks with low variation based on attributes. Global variation could be high, but there is a need that cases with identical attributes must be produced with little variation. If this is true, we could analyze different sets of records divided by attributes (every set of record has its own attributes and every records in set have the same attributes). The sets will have some execution times and time of every set will be ordered ascending. We will get something like this for some set of records with the same attribute. For example, there are twenty records for attributes A, B, C: 100s, 101s, 102s, etc., 110s, 112s, etc. 140, etc..If we suppose low variations of executions for the same attributes, first records are the real execution time whereas others are probably affected by waiting time.

Half measurement is not only burden that negatively affects accuracy of prediction but it can be also advantage. Measurement devices are also quite expensive so every saving counts.

IV. MANUFACTURING COMPANY

The manufacturing company is specialized in door production. Uniqueness of door is characterized by their attributes (about twenty) and based on these attributes, different operations are executed. Doors have different material, size, weight, different corner and edge types, different handle and glasses, etc. Every door has its ID and it can be modeled as a case. Doors are mainly manufactured in machines (tasks). Some machines work in parallel; some machines are bound to several tasks, thus these machines must be treated as resources, because machine could be down or working. People are working with machines or in manual workplaces. Routing probabilities are 100% accurate, because doors with specific attributes must be manufactured only by a specific machine and with specific settings.

Resources are quite predictable, because they work on shifts and they are always available and planned several days ahead. The only unknown parameter is execution time of tasks that depends on case attributes – every case is modeled as one door, so case attributes are door parameters. Door parameters are known at the beginning of the process and are constant, so there is no need to build several classifiers for several periods of case execution [10]. Execution time also depends on people work rate, work queue and error rate (especially in manual workplaces), but this is issue beyond the paper.

Context-based predicting the execution time of tasks can help with several issues quite precisely. First, it is possible to decrease storage spaces, because they could plan execution order of cases in order to decrease waiting times. Our prediction decreases variances of execution time and thus logistics can have methods to plan storage spaces with better results when there is a low variance. They will also know if some doors will be probably late and for example they can respond to that changing priorities, resource allocation, etc. Another important issue is the analysis. Managers could measure which door types take long time to be produced and based on that, they can calculate their price more precisely. For logistics, execution time is not as much important as influence of variance of execution time. It is possible to measure which door types (based on parameters) have high variance. Process engineers can focus on those door types and try to find out the cause of high variance, or produce them only in situations (if it is possible to wait) when variance is not such important issue.

A. *Usage in Company*

Because our manufacturing company has quite a lot dependence on door attributes, it is important to discover them to better plan production. Order of different doors for production is also important, because some machines could produce some door types faster, some more slowly and we need to better balance the product flow.

V. Prediction of Tasks Execution Time Using Classification

The time deviation is sometimes high, but it can be decreased by data mining techniques. Thus, it is useful to examine data and find relationships between case parameters and execution time for each task in the process. This can be solved as a classification problem, where case parameters are considered as input attributes and execution time is considered as the target attribute.

A. *Classification and Prediction Models*

There are many kinds of classification models; every model has its advantages and disadvantages based on properties of data used for classification. Our problem is rather prediction than classification, but both issues are related and many models support both of them. One common definition is that prediction predicts future and classification works as pattern recognition. Other definition is that prediction works with numerical target attributes and classification with categorical target attributes. Prediction can be transformed to classification by transforming target attribute from numerical to categorical, where categories are intervals that covers whole domain.

In our case, we have 18 case attributes and one numerical target attribute. All case attributes are categorical. Even through there are also some of them numerical (width, height), but they are standardized to only few distinct values, so they can be considered as numerical or categorical depending on requirements of classification or prediction model. It is more difficult for prediction (even in our case) that target attribute varies even for cases with the same values of input attributes. This is typical for execution time, because work is performed not only by machines, but also by people and people do not often work in coherent speed.

High variability of door types is another problem. In our manufacturing company, it is possible to make millions kinds of doors, which causes problem in prediction, because it is difficult to obtain enough examples for prediction. Attributes can also contain high number of distinct values which correspond with high variability of door type (this is a problem for neural network classification).

In the next section, some prediction models will be described and its applicability is discussed.

1) *Neural Network*

We have tested Neural Network approach, but results have not been satisfactory. Neural Network was not able to learn. It was caused by the high number of input neurons - 303. Every categorical column had to be transformed into new columns. Every distinct value of that column created a new column, which holds 1 or 0 value. For example, the column corner has four distinct values – left, right, top, and bottom. It creates four new columns that can acquire value 1 only once for a row (for the columns that belong to one categorical column). That transformation was necessary, because neural network can handle only numerical attributes. Target attribute was divided into several intervals and every interval was modeled as a single output neuron.

We think that network was not able to learn because of high number of inputs compared to number of training examples and mainly because of the output variability of (even identical training examples had little different outputs). Thus, we think network is not sufficient for our problem because of high number of categorical attributes and variability of the target attribute.

2) *K-Nearest Neighbour (KNN)*

The method is based on a simple idea of finding several examples from training set closest to an input pattern. We simply computed number of differences between training example and input pattern. These differences (0 or 1, equals or not equals) were weighted. Weight of each attribute was computed by the same method described below by the regression tree. Higher weight means that attribute has higher influence on execution time and it is considered more important. Then twenty nearest examples were given and mean, minimum, maximum and deviation of time was computed (we measured only mean, but deviation is also important in simulation and it is a good indicator of supposed reliability of prediction).

Results (Figure 4) were quite satisfactory (there is only subset of real workplaces). We have compared prediction to a simple algorithm – prediction based on mean of all execution times. The simplest predictor is the predictor that assumes mean value for every example. Differences in table are mean of all differences between real values and predicted values for every tested example. We have run the test with 600 examples and we have compared them to a dataset that contained approx. 10-20 thousands records for every workplace. Rating was computed as a ratio between difference computed by the algorithm and difference computed by mean. Thus, the result 0.5 means that we have decreased the variance of execution time of task by about 50%.

Figure 4 shows that some results are satisfactory, others are worse. For example, ratio of workplace A seems to be good, Workplace C is not as good as Workspace A. Nevertheless, it is not the problem related to the method, execution time does not rely only on attributes. It is for example the case of workplaces C, which perform packaging and that type of work is naturally quite independent of door types. .Workplace A is a machine that does not depend on resource skills, workplace B is a workplace with dependence on resource skills and workplace C is a manual workplace (packaging) that does not depend so much on door type, but on resource performance

3) *Regression Tree*

Decision tree is a popular model. It is simple, readable by human and quite fast. Precision has not been as satisfactory as results given by the K-Nearest Neighbor classifier. However, the classification speed is several hundred times faster. Regression tree is a decision tree with numerical target value. Nodes contain information about mean, minimum, maximum and deviation of predicted value. Learning algorithm is similar to decision tree, but selection of split nodes differs. We have numerical target attribute, therefore, algorithm can be as follows:

```
Input: A table, which contains a numerical target attribute.
Output: Decision powers of all attributes.
1.  For each column $C$
2.      For each distinct value $v_i$ of column $C$
3.          Take all $n$ target values $t_i$ of column grouped
            by current distinct value and compute their
            deviation $\sigma_i$.
4.      Count the decision power of the column as:
        $DP(C) = 1 / (\Sigma\sigma_i / n)$, i.e., mean of all deviations.
```

Algorithm 1: Regression Trees

This algorithm is similar to entropy computation, which is computed for categorical target values. The deviation is closed to entropy because lower deviation points to better decision power. Computing of deviation can be also weighted by count of rows related to groups divided by distinct values of column – distinct value with more rows should be more important. We have tried both approaches, but no significant precision difference was observed, even maybe precision has been a bit lower. Algorithm described above works similar to ID3 algorithm. C4.5 algorithm has been also tried, however, no significant difference has been found. Post-pruning was based on removing nodes with low row count (every node corresponds to a subset of the whole data set), because nodes with low row count are not representative.

Regression Tree has had worse precision compared to K-Nearest-Neighbor (about 1.2 – 1.3 times worse), but it has had also several advantages. It is more readable to human and it can be used to examine some properties of tasks – for example, which combination of attributes affects execution time positively or negatively or which combinations of attributes have little ratio of prediction – that is represented by deviation of target values corresponding to some node of tree.

4) *Regression Tree Forest*

Regression Tree Forest is based on several Regression Trees. Obvious example can be the Random Forest. The Random Forest creates many decision trees (more than one hundred) using classic (ID3 or C.45) algorithm with several differences:

1. Every tree randomly selects subset of rows from a training set (about 2/3).
2. Every tree randomly selects subset of attribute columns (about 2/3).
3. Every tree is not pruned and full-grown.
4. Predictions are made by voting of all trees by computing mean.

It is known that Random Forest is a very precise model and it is still quite fast, because it is semantically similar to K-Nearest-Neighbor algorithm. However, learning time is quite long (it requires more than one hundred trees), we have found it not suitable for real-time decision support. However, we have tried some trade-of between Random Forest and normal Decision Tree. We created several (about ten) trees and enforced different first splitting column for every tree. Enforced columns were ordered by their decision power. Thus, first tree root node begins with the first (best) column; second tree root node begins with second column, etc. In addition, every tree randomly selects 70% of dataset and 70% decision attributes as it is used in Random Forest algorithm. Trees were pruned (opposite to Random Forest, which is not pruned) to about 10 rows in a node.

It should be stressed out that in normal Random Forest, result is computed by mean of all tree results. We have selected the best tree result by looking to the deviation of tree node. Best prediction could be measured by deviation of particular rows covered by a tree node. Node with lowest deviation wins. This rule was necessary because mean of all tree votes gave very unsatisfactory results – mainly because we have had only low count of trees compared to Random Forest.

Then, we designed another improvement – tree result (mean and deviation) was not computed only by looking at the leaf node but also the parent node is taken into account. We have computed mean by both nodes mean values weighted by their deviation (if a child node had much better result – low deviation – than parent, its result will have bigger vote). That improvement has also been tested in a single ordinary Regression Tree and it increased precision too, but only slightly.

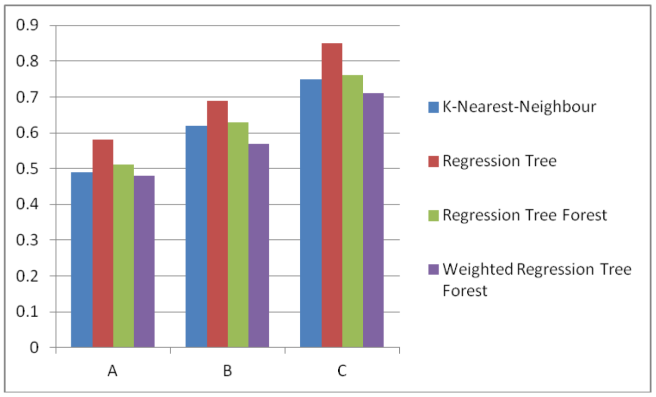

Figure 4. Experiments. Four methods were used on three workplaces. Note that a higher column means lower precision.

Our Tree Forest significantly improved accuracy of classifier, the ratio was only about 1.05 times worse compared to K-Nearest-Neighbor, however, it was the order of magnitude faster than K-Nearest-Neighbor, usability of which could be problematic in real time monitoring for all tasks due to performance issues. Similar results can be explained, because random forest works similar to K-Nearest-Neighbor. It returns items that are close (by attributes) to a predicted item, but it uses tree searching instead of searching in the whole table.

B. *Execution Time and Resources*

There is a little problem with resources. The resource information can be treated as a normal case attribute, because it has impact on execution time of task, but there can be some difficulties. For example, if we allow decision tree to build tree using resource attribute, final leaf will contain only records that were executed only by that resource. This could cause problems because sometimes it is better to look for more examples, even from another resource. However, if we do not have such training examples and resource performance does not differ too much from other resources, it is useful to look also to another resource records and consider them.

The second problem is related to dynamic changes. Even if the process is the same (e.g., technological process), workers performance changes over time. More experienced workers may be faster, thus our algorithm should be prepared for dealing with that kind of issues. We recommend the following method, which slightly improved prediction in our manufacturing company.

Suppose K-Nearest-Neighbour or Regression Tree (or Forest) classifier. All that classifiers could be implemented to return set of records rather than final prediction (mean and deviation). The result (mean, deviation) could be implemented over those records, but with different weights. First, records that belong to the resource, performance time of which is now predicting, should have bigger weight (for example two times higher) than other records. Second, these records (of our resource) should be considered in the time plan. Newest records should have also higher weights (for example two times higher than the oldest). Why it is not possible to take into account time plan also for other records (other resources)? Because it is very difficult to know about them enough information in order to take into account their improvement and skills compared to our resource. This could be issue to another paper.

C. *Computing Variance*

As we mentioned, low variance is important for good process of planning and material flow. It is useful to know variance of a task based on its attributes. Our tested method returns the set of examples and then computes means from them. It is not difficult to compute variance as well. But there can be a problem with different methods that do not return a set of examples, but only the final decision (not a regression tree problem, because leaves contains mean and also variance information even if algorithm is not built to return set of examples). For example, it can be problem with Neural Network, which is not built to return examples at all.

D. *Test on Validation Data*

Previous results were tested on training dataset. We have done another test using test dataset (about 20% of whole data) to prevent overfitting. Results were very similar to previous tests; therefore, we did not include them here. It is because methods like KNN and Regression Tree are quite robust to overfitting. The second reason is related to data – there is some variance even for records with the same attributes. So overfitting is not an issue here.

E. *Summary*

We have tested three tasks: One machine with little human interaction, second machine with manual work and third packaging with little dependence on door type, but with dependence on resource. As it was presented, Regression tree is always less precise than other methods, while Regression Tree Forest is as precise as K-Nearest-Neighbour, because it is like the optimized K-Nearest-Neighbour. Last method was the weighted Regression Tree Forest. As it has been shown, weighting on workplace A did not improve result at all, because machine works independent on resources and time (it does not learn to work faster). In Workplaces B and C, there was some improvement. Workplace C had the worst results, because packaging is not dependent on door type too much, but it is dependent on resource – we can see that weighting slightly improved performance.

VI. PREDICTION OF EXECUTION TIME OF TASKS USING ASSOCIATION RULES

In some cases, there is a need to process non-relational data because sometimes the sizes of various cases can be different. Their main difference from classic relational data is the fact that various records can contain various counts of values (case parameters). To predict execution time of a process, it is suitable to use the association rule based classification because data are similar to a transactional database. For our task, we will consider each case as a transaction.

A. Mining Association Rules

Association rules were first introduced by Agrawal et al. [18]. Mining association rules was primarily designed for usage in transactional data. Therefore, it is not any problem with discovering association rules from this kind of data. A lot of algorithms for mining association rules in transactional data have been developed. The Apriori algorithm [19] is probably the most famous of them because of its simplicity. On the other hand, the FP-Growth algorithm [20] proved to be much more efficient than the Apriori algorithm.

Association rules are most frequently used in market domain, typically for market basket analysis, where transactional databases are used. Here, the goal of mining association rules is to find rules of a form A⇒B where A and B are sets of items. If this association rule is found, it is usually interpreted as: “If a transaction contains a set of items A, it is likely to contain a set of items B”.

A formal description of the association rule mining problem is specified as follows. Let I = {i1, i2, …, in} be a set of items, which can be contained in a transaction. Let T = {t1, t2, …, tm} be a set of all transactions, where each transaction ti = {ii1, ii2, …, iik} is a set of items, where each item iij $\in$ I (for i $\in\langle 1,m\rangle$ and j$\in\langle 1,k\rangle$). If A is a set of items, a transaction t contains A in case that A $\subseteq$ t. Then, an association rule is defined as an implication of the form A⇒B, where A, B $\subset$ I are sets of items, which are called itemsets. These sets must be disjoint. An itemset that contains k items will be called k-itemset.

Potential usefulness of a rule is usually expressed by means of two measures – support and confidence. The rule A⇒B has a support s, if s% of transactions contains both itemsets A and B. It represents the probability of occurrence of the rule in the database. This probability can be expressed as

$$support\ (A \Rightarrow B) = P\ (A \cup B) \qquad (1)$$

Confidence of the rule represents the strength of implication in the rule. It is the conditional probability that a transaction contains the set B provided that it contains the set A. This is expressed as

$$confidence(A \Rightarrow B) = P(B|A) = P(A \cup B)/P(A) \qquad (2)$$

For each association rules mining task, a value of minimal required support and confidence must be specified. If the condition of minimal support and confidence is satisfied, the association rule is called a strong association rule. A frequent set is an itemset satisfying user-defined minimum support and strong rules are generated from frequent itemsets that satisfy also user-defined minimum confidence.

B. Association Rule Based Classification

The model described above can be also adapted for classification. The association rule based method was originally designed for classification of text documents into a set of predefined categories. Each text document is represented as a transaction – a set of words (terms), which occur together in the corresponding text document. Generally, a transaction is defined as a set of items. Therefore, the only required information is the occurrence of the term in a document.

Association rule based classification method was first introduced in [21]. The main advantage of it is that it provides a human understandable classification model in a form of association rules and good accuracy of classification. The Apriori-based algorithm is used to generate association rules.

The next method called CMAR [22] was based on a well-known FP-growth method for mining association rules, which is significantly faster than Apriori-based algorithms.

The method described in this paper is a modification of association rule based classification method designed for text classification. The original method described in [23] works with text documents represented as transactions (set of terms, which occur in the document). In the training phase, association rules are discovered from these transactions for each class separately by use of the Apriori algorithm.

For this classification method, we are focused only on a special form of association rules, which is usable for classification. The required form of a rule, which is a result of the training phase, is the following:

$$term_1 \wedge term_2 \wedge \dots \wedge term_n \Rightarrow Cat, \qquad (3)$$

where the antecedent of a rule contains a set of terms, which frequently occur together in documents that belong to category (class) Cat, which is contained in the consequent of an association rule.

We can see that the task of training phase of the classification method is to obtain a set of association rules for each category. This set of rules is forming the classifier.

While the set of categories is predefined and there is a set of documents belonging to each category, we can obtain a set of association rules separately for each category. For each category, a set of frequent itemsets is found with use of Apriori algorithm in a set of documents belonging to the corresponding category. Each frequent itemset is then associated with that category.

That is why the method is called ARC-BC (Association Rule Classification – By Category). This property allows to perform so-called multiple-class classification, because there can exist rules with the same antecedent and different category in the consequent. If it is necessary to assign only one category for each document, we have to decide according to the value of support or confidence of a rule. The association rules with lower value of support/confidence are omitted from the classifier.

If the set of rules is generated, it very often contains very high number of similar association rules. To reduce their quantity, pruning techniques can be used. The task is to find association rules that are more general and have higher confidence.

If we have a suitable number of rules for each category, we can use these rules to classify new objects (records, documents, etc.). We have to obtain a set of terms representing the new object. If there is an association rule that contains the same set of terms, the corresponding category is assigned to the object.

Usually, more than one association rule (for more than one category) is found for a classified document. It is necessary to set a dominance factor, which is counted as a sum of association rules' confidences. This allows getting a most dominant category or k most dominant categories for a document.

In [24], this method was adapted to use it for classification of classical relational based data but the experiments presented showed that it is not very accurate if the table contains more quantitative attributes, which must be discretized before association rules are going to be discovered. In our process mining task, discretization of input data is not required.

C. Usage in Process Mining

The main difference between our process mining task and the process described above is the target attribute, because we have to predict a numeric value of processing time in our data mining task. The task must be transformed into a classification task. The only possibility to do it is to discretize the target attribute. Regarding the discretization, we have to make two decisions – we have to choose the discretization method and the size of intervals created from the values of the target attribute. This choice depends primarily on the end user's requirements.

Sometimes there may be a need to make some post processing steps after the training phase of the method because the same frequent itemset can be obtained for different categories by this classification method. There are two possible post processing tasks resulting from this fact:

If the same frequent itemset is obtained for two adjacent categories (intervals in the antecedent of a rule), these two (or more) categories can be joined to form one association rule.

If this appears for two or more intervals that are not adjacent, we can omit the rule with lower value of support, if it is significantly lower. If the difference between two support values is not very high, we should keep all those association rules in the result of the training phase.

Discussion about this solution and presentation of some other interesting properties of this method will be contained in the experimental part of this paper.

D. Experiments

We have made some experiments with a real dataset from a door producer. The task was to predict time needed to make a door from the set of input attributes, such as material, size, weight, etc. The dataset consists of 17 input attributes and its size is 10000 records.

It was necessary to discretize the time attribute to apply the association rule based classification - the attribute was discretized into three categories (intervals), which represent low, standard and high time needed to produce the door.

1) *Classification Accuracy*

Our dataset is a relational table and, therefore, it is possible to compare association rule based classification with other classification methods.

The value of classification accuracy is between 75% and 80%, depending on values of minimum support and confidence thresholds given by the user. This value of accuracy is comparable with other classification methods such as naïve Bayesian classifier, decision trees or support vector machines. This leads us to a conclusion that our method is suitable to predict time also in data with variable number of attributes.

2) *Examples of Association Rules*

As the association rule based classifier provides a user understandable model, it is also possible to analyze the association rules for individual categories. Probably the most interesting for the user will be the category, which represents high produce time.

We can mention an example of association rule obtained by our method: door_construction = type_1 $\wedge$ edge_D=CH001 $\Rightarrow$ high_time.

This kind of association rule helps producer to plan his production and to find the reason of delays during the production process. We have obtained about 30 association rules for the category representing "high production time". Similar count of association rules has been obtained also for other categories.

It is also interesting to analyze association rules obtained for more classification categories. This denotes for higher dispersion of time value. This fact also means that attributes contained in those rules have less influence on the time of production. If these association rules are joint into one rule, we have association rules of the form from the following example: hardness = 1 $\wedge$ filling_type = DTA $\Rightarrow$ high_time $\wedge$ avg_time $\wedge$ low_time.

At the end, we can take only interesting associations (only those with class high_time or low_time) and then select all records that contain these attributes and compute mean and variance for them. For example: type_1 $\wedge$ edge_D = CA003 $\Rightarrow$ 230s ± 100s.

VII. UNMEASURED PROCESSES

Unmeasured processes are processes with known process model, however, only the length of execution of the whole process is known. This case can be related to many companies, because measurement of every process step is expensive. But even if we do not know the exact time for every task, we are able to discover some of them if enough data is available.

Unmeasured processes could be static or dynamic. Static processes are processes that have always the same tasks in their process model. For dynamic processes, the process model is built during the runtime based on attributes of a process instance (case).

Analyzing hidden processes could be easily topic for whole new paper, thus we will describe only situations related to our manufactory company. But our experience should be applicable for many other companies with the similar problem.

We give theory only for dynamic processes because there is one problem with static ones – if every process instance is the same, we cannot compute what is inside the process, because it is the changes what allow us to find something valuable. On the other hand, static processes are more predictable and they could be treated as atomic task.

A. *Sequential Dynamic Processes*

Sequential dynamic processes are processes with sequential flow but every process instance contains different tasks. One process instance could contain tasks A, B, F, H, another one could contain C, E, I. We know duration of the whole process instance and names of executed tasks (and their order, if it is important). So our information can look as follows:

A, B, F, H	=	213s,
C, E, I	=	170s,
etc.		

Our objective is to assign average time duration to every task in order to best fit the execution time of the whole process. It is obvious that total execution time is the sum of lengths of all tasks in the case. If A = 100s, B = 60s, F = 20s and H = 17s, the case with tasks A, B, F, H is supposed to be executed after (A+B+F+H) 197 seconds. How much close we will be to real time (213s) depends on quantity (and quality) of measured data, total number of possible different tasks (size of problem space) and nature of process. If a process is predictable (for example machine) then result will be close to real values. If a process is not predictable (mainly manual labor) then the result will be far from real values, but it can be still usable. If a process is exact, it is possible to solve the problem with linear equations but this is not common for most situations. Even if there is a machine with quite predictable speed, its swiftness could be dependent on some case attributes (which are or are not available) and even if not, there will be still some little variance – for example machine must be supplied by material, however, the material flow can differ from machine speed. Analyzing unmeasured dynamic processes, which is also highly dependent on attributes, is not an easy issue to solve. First, we will describe how to analyze process, which is not dependent on attributes.

We have created quite simple algorithm, which was able to compute satisfactory result quite fast. Our algorithm is an iterative computation based on heuristics. Initially, we generate candidate solution to heavily speed up iteration. We make the following computation:

```
Input: Dataset, which contains all process instances
Output: List of predicted times for all tasks
1.  For each record (process instance I, which contains of
    N tasks)
2.      Compute execution time for every task in case as:
3.          t = Execution time (I) / N
4.      Add the value of t to time collection of the task.
5.  Compute average from these times for every task.
```

Algorithm 2: Initial Candidates

Example:
A, B, D => 210s
A, C, D => 240s
210 / 3 = 70. A = {70}, B = {70}, D = {70}
240 / 3 = 80, A = {70, 80}, C = {80}, D = {70, 80}
Averages: A = 75, B = 70, C = 80, D = 75

Second step is the Iterative Algorithm is following:

```
Input: Dataset, which contains all process instances
Output: Improved list of predicted times for all tasks
1.  Randomly select one task T.
2.  For every record $r_i$ (process instance):
3.      If $r_i$ contains T then continue
                   else go to next record (2).
4.      Count difference between real time and predicted
        time as: $diff_i = real – predicted$.
5.  $diff = diff / number of records containing T$
6.  $task time = task time + diff * learning coefficient$
7.  Compute error E as:
    $E = \Sigma diff_i$ (sum of all differences)
8.  Repeat algorithm until error decreases.
```

Algorithm 3: Iterative Algorithm

We have used the value 0.2 as a learning coefficient. Learning coefficient is a common method for iterative computing, because we do not want to converge to suboptimal solution. Good example is the perceptron learning algorithm, which was inspiration for our method.

We have tested our method in manufacturing company at a manual workplace. This workplace always makes several different tasks on doors based on customer demands. Tasks are known, however, only time of whole process instance (all tasks on one door) is available. We had about 44 thousands of process instances and about 5 thousands of different tasks. This is a big number but most of them occurred only once or twice. Only about two hundreds of tasks appeared frequently.

Management previously used simple average time for every process instance regardless what tasks were included in the subprocess. Similar method was used as reference solution in previous sections, thus we have used it again. Its error was computed in the same way as it was mentioned in in step 2 of previous algorithm. The reference error is obtained as a sum of all absolute values of difference (|real – average|) of times for all records.

Error for average time was 131 seconds (average time was 354 seconds). After application of the first algorithm (computation of initial candidate), we lowered error to 84 seconds. After few seconds run of the algorithm (iteration), error was decreased to 71 seconds. Thus, initial error was 354 ± 131 and we reduced it to 354 ± 71 seconds. This is a satisfactory result and we do not suppose it could be much more improved because of natural unpredictability of manual processes.

1) Setup Time

Every process may have a setup time. That hidden task must be done for every process instance, for example the administration or reading the line information from product and etc. It may be better to use the normative information (measured by standardizer). The problem is that the algorithm itself will provide the same error results regardless what the set up time is (because the setup time affects all the process instances in the same way, this is the problem for static processes as we have described it at the beginning of this section). But if you want to use task times not only for prediction then it is better to measure setup time by standardizer and add it to every process instance and then compute more realistic task times.

2) Average Time for Task

An intuitive way is assigning all tasks an average value. This can be done by dividing all process instances by number of tasks in it (if there is setup time, we must subtract it). Then, we will count average value from these numbers. The prediction is now counted as: Total time = setup time + number of tasks * average time for task. However, result of this method was 145 seconds, which is even worse than our base error (131). This is the reason why we did not include this method in overview. But it could work for problems with similar task times, which was not our case.

3) Genetic Algorithm

We have also tested a genetic algorithm with one point crossover, tournament selection, mutation and 50 individuals in generation. Every individual had a vector that represents times of particular tasks. Result of the genetic algorithm was not better than the previous method (also not worse), but we were testing it repeatedly for several hours. Previous algorithm was able to compute it in about half minute. Thus, our method is good enough and very fast, with results comparable to genetic algorithms.

We have tried the same genetic algorithm with starting individuals close to solution that was found by previous iterative algorithm, but result was only slightly better (error value was 69 seconds).

4) K-Nearest Neighbor

K-nearest neighbor is the good classifier as we used it previously with satisfactory results. Our result was also satisfying. It reduced error to 73 seconds (from 131s), which is very close to our previous results. However, it has a high computational demand, which can cause problem while using it in real time monitoring.

Because of different lengths of vectors (process may contain any number of tasks), there are multiple ways how to implement similarity. We have chosen this one:

At first, all close vectors are found. Those vectors must have the same length and must not differ less than in one task. But if the vector length is smaller than 3, the vectors have to be completely similar. We have got a result - collection of pair – time (that belong to vector) and weight. The similar vectors have weight equal to 1; slightly different vectors have weight equal to 0.7.

If the result collection has too few items (e.g., less than 10), it is used average time of all records instead. This last rule highly depends on nature of the process. In highly predictable process, variance is too low, thus it is possible to compute result from a small number of examples. But processes with relatively high variance will need more than few items.

We have tried several different settings of this method, but all settings led to similar results. It seems that error about 70 seconds is close to global optimal solution of the problem.

5) Process with Attributes

To gain better result, we need more information. Process attributes are suitable candidates. Process attributes can be some descriptive attributes of the process or resources involved in the process. Attributes can affect execution time of the process in the same way as affected it in previous sections. The only difference is that we also have different set of tasks in different process instances, thus ordinary classification methods are going to fail, but not while combining them with some previously mentioned methods (classification or association rules). Let us suppose, we have computed average time of every task (with some error, of course). Predicted time of process instance is a sum of average time of its tasks, as we described earlier. So we have two times – real time and predicted time. We need to compute their ratio.

$$ratio = real\ time\ /\ predicted\ timed \qquad (4)$$

The ratio is now the target attribute to predict. Now, we can use any classifier (or association rules), as we described them in Sections V and VI. In the testing phase, we can obtain ratio of the process instance based on attributes. Then, we will compute predicted time based on sum of average times of its tasks. Finally, we will multiply ratio with obtained time.

We have tried the decision tree forest and difference error of 66 seconds (result of prediction without attributes is 71 seconds) was obtained. That is not an excellent result but it is usable. Problem consists in the fact that the process of computing average times of task is itself inaccurate, thus the result of classification cannot be so accurate as well. But the total result – error of 66 seconds from initial 131 seconds is still a useful result.

6) Multiple resources in process

If there is only one resource, it is possible to use resource as ordinary attribute. But what if there were multiple resources working on different tasks? Or, what if it is known how effective the resource is? We can use similar approach that was described in Sections V and VI, only with one difference. We cannot multiply classifier result with the total process time, but we must first multiply resource efficiency with its task time. For example, John did Task A and B and Jane did task D. Times are A = 2 min, B = 3 min, D = 1 min and worker efficiency is John = 0.7 Jane = 1.1. Then total time = Classifier ratio * (2 * 0.7 + 3 * 0.7 + 1 * 1.1).

It could be quite hard to compute workers efficiency from that kind of data, thus it could be better solution to gain it differently (from some data with more precise information or from a standardizer). Of course, worker could be more skilled in one process and less skilled in another, but manager's experience tells (for manual jobs without high qualification), that if worker is slow in process A, it will be probably slow in process B as well.

7) *Computing Variance*

As it was said previously, variance is very important for planning– mainly for material flow and overall planning. High variance causes high inventory and there is a problem with synchronization, which can cause deviation from desired plan. Variance could by computed using this approach. The main idea behind it is that we have set of process instances (records) that contain that information – set of tasks, and real time of process execution and finally predicted time (previously computed by some of our methods). Variance for a task is computed as follows. At first, all records containing these tasks are selected.

Then, we need an average value and a current value, which is a need for variance computation. We do not know what the average value is, but we can suppose that predicted value of that record is close to average value of the process instance because this is what our methods needed to compute. Then, current value is the real value and (real time – predicted time)2 is a base assumption.

```
Input: Dataset containing all process instances.
Output: Values of difference for each task.
1.  For each task T
2.      For each record r containing T
3.          $diff_T = diff_T + (real – predicted)^2$
4.  $diff_T = diff_T$ / number of records containing T
```

Algorithm 4: Counting the values of difference

Note that variance computation is only an estimation, not an exact mathematical calculation. If a task causes high deviation, then the records of the task will also cause high deviation (difference between real and predicted time). However, if a task with high deviation is present many times with task with low deviation, their deviations will be average, so this is only the estimation, which is possible to be wrong and it highly depends on tasks, which are in the process instances.

8) *Summary of Methods*

Figure 5 is an overview of used methods. We can see that precision of all methods are nearly similar, except that genetic algorithm and K-Nearest-Neighbor takes much computing time – genetic algorithm has very long learning time and KNN very long testing time. We can also see that classification using attributes slightly improved error rate.

9) *Validation on test data*

We did also validation on test data (Data Set was divided by 99% for train data, 1% for test validation data). This unbalanced distribution was caused by fact that every day some new operation that was not in previous data occurred. So if we divided our data for example by 80% and 20%, results were very bad. There is a high need to learn system continuously in real production to better estimate time of brand new tasks. This is possible, because our algorithm needs about one minute to run.

We tested 99% of train data with fifty rounds of cross validation. Result was slightly worse than error in training data (about 1-3 minute to error of every method) so we did not include it here.

But we also include one little thing – in real prediction, every day a new operation with unknown time occurred. We have assigned it average time of all known tasks times.

B. *General Dynamic Processes*

General dynamic processes could include parallelism and every process instance could be a different process with different tasks. Resolving task performance times for general processes is more complicated than for sequential processes.

Example of some process instance with parallelism (symbol || means parallelism, symbol + serial execution): A + B + ((C + D) || (E + F)) + J.

This means that A and B are executed serially, then two parallel branches are executed – first C and then D, second E and then F and after waiting for both branches task J follows. Another example of process instance could look like that: A + (C || (E + F + B)) + D + D.

We did not deal with that kind of problem in our manufacturing company; however, we would like to propose some ideas for future research in this area. Because general dynamic process could be any process, every process instance has to contain process definition (or log with parallelism – see below), not only tasks. If only tasks are available, we may use Process Discovery first [6]. This could work in some cases, but we will first focus on processes with known model.

Let us suppose we know average times of all tasks. Then, the question is, how to compute final process instance time? Two examples above show a task execution log enhanced with parallel execution. What is the difference between normal log and log with parallelism? Normal log for both examples should look like: A, B, C, E, D, F, J and A, C, E, F, B, D, D.

This log contains information about executed tasks in order to their completion time (or start time). Using this, we do not know exactly, which tasks were executed in parallel and which of them sequentially, thus we cannot solve anything. If there is only a sequential log, process model must be available (either for every process instance or globally for all process instances). What is difference between process model and log with parallelism? Look at the second example: A + (C || (E + F + B)) + D + D.

This process instance should be executed by many different process models – for example, two final tasks D could be in process instance log, where the task D could be in a loop that allows repetitive execution or it could be always two serial tasks D (for whatever reason).

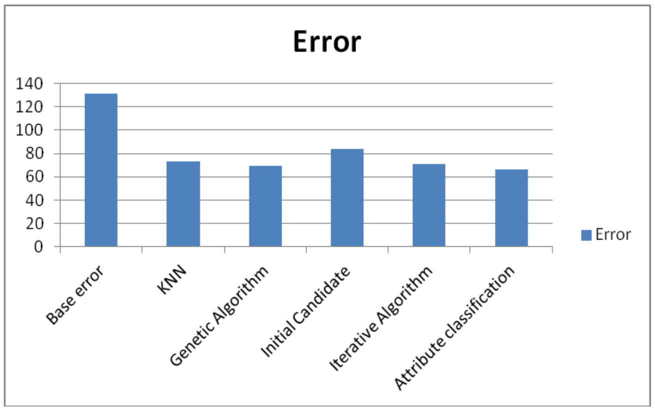

Figure 5. Results of different methods for dynamic sequential processes. Base error is computed as an error if we use average mean value from whole data set for all predictions.

How can we compute total estimated process instance time if we know average times for all tasks? If we have a log with parallelism, it is not so difficult. We must simulate this dynamic process as it is – if there is a sequential execution, we will be adding performance times. If there is a parallel execution, we will be able to continue after slower branch is completed. For example, first process (A = 2 min, B = 3 min, C = 4 min, D = 1 min, E = 1 min, F = 2 min, J = 5 min): A + B + ((C + D) ‖ (E + F)) + J.Total = 2 + 3 + Max(4 + 1, 1 + 2) + 5 = 15 min.

Function Max returns maximum from two input numbers. If there is a more complicated process, we must use recursion. But what if we know a process model and a sequential log? We can use log replay as it is described in [6] in the section about Conformance Checking.

1) How to find solution?

How to find average times for huge set of process instances? It is important that every process instance is at least a bit different as we discussed it earlier. Static processes are always the same and there are low chances to analyze what is inside the process.

Because of the complexity of the problem, we suggest to use Genetic Algorithm. Solution can be coded as a vector of real numbers. For example, process instances could contain four tasks – A, B, C, D. Thus, we will have vector with four real numbers. Fitness of the solution is an evaluation of all process instances and computing error. Error should be computed in the same way as in Sequential Dynamic Processes – error = | real time – predicted time|. After that, genetic algorithm setting continues (selection, mutation, crossover. number of individuals in generation, static / steady state, etc.). We cannot say what setting will be the best because it highly depends on current process instances.

We believe that the genetic algorithm should be able to find average times of tasks to find suboptimal solution. But closer experiments are beyond the scope of this paper.

2) Problem with Process Discovery

Process Discovery is able to find process model from logs [6][7]. Problem of this solution is that it does not distinguish between serial execution with arbitrary order and parallel execution. For example, let us have two logs: A, B and B, A.

Most Process Discovery algorithms (as Alpha Algorithm [7]) see it as a parallel execution, because it does not depend on the order. This should be acceptable when we are analysing log to discover some usable process model that represents some probably possible executions of process, but it is not suitable for our time prediction. Sequential and parallel executions are evaluated differently. If A is 2 minutes and B is 3 minutes, than sequential execution takes 5 minutes and parallel 3 minutes. Administrative processes usually also allow this parallel execution really in parallel (if electronic documents are used). In manufactory, products cannot be produced on both in task A and task B in the same time, If parallelism is discovered then it only means that it does not depend on order of task. But there is another problem – what if there is some material in process that will be mounted on product later during processing? Now, it is possible that there is a parallel work, which indicates a problem.

However, Process Discovery could still be usable if we know which resource executed which task. If task A and B are executed by one resource, we know that even if Process Discovery says it is parallel execution, one resource must execute it sequentially.

Note that Process Mining discovers global process model for all process instances with OR routing branches, so log replay must be used to merge process model and log as we mentioned above, see Conformance Checking [6].

VIII. ANALYSING ERRORS

Primary analysis of execution time is the main focus of this paper but we can also analyse errors. Errors can be also dependent on attributes. Errors are an important part of simulation. We cannot simulate errors with very low probability but we must simulate operational errors like defective products. Error situations must be sometimes analysed separately from the time analysis (error in products must not be involved in time computation), because errors are mostly treated differently – for example defective products will be sent to repair (another task, similarity with Decision Analysis (Section III), or thrown away.

1) Classification methods

Classification methods can be easily transformed to predict errors instead of time. Error will be used as a target attribute of record for classification (0 – no error, 1 – error). After that, classifier can predict error for given attributes – for example it returns value 0.07, it means there is 7% probability of error in that task.

A. Association methods

It is also possible to use association rule based classification to predict errors. There are two classification categories in this task – 0 (“no errors”) and 1 (“error”). It is not difficult to obtain some association rules and classify the cases but the main problem is the fact that the training data for the "error" category is much smaller than the "no errors" category because it is expected that most of products will be made without errors and no cases will be classified to "error" category.

Therefore, we will concentrate only on association rules obtained for the "error" category. Deeper analysis of this set of association rules can show us some interesting properties of cases, which lead to some error.

To use association rules for classification, some complex post-processing of association rules should be designed and implemented. This is one of the issues to be solved in the future research.

B. *Unmeasured process*

Error could happen in unmeasured processes. There could be many settings of how error should be handled. Process should stop immediately when error occurred, or it could run to its end and then all errors are resolved. Other information is related to error itself – if we know, which task contained error and which one did not – in this situation we know only that the process instance finished in error state. First situation will result in more precise probabilities and it is also quite simple to compute – if there are no attribute dependencies, we could only compute successful and unsuccessful (error) execution and compute ratio.

Second situation is more complicated. We know that error occurred somewhere in process, but we do not know where it is exactly located (in which tasks). However, it could be easily solved by a genetic algorithm. Vector of tasks error probabilities is a possible solution. Fitness of this solution could be computed this way. We will resolve all process instances using random generator – process instance will be replayed and error will be randomly generated using error probabilities for tasks. For example:

Assume Serial Process: ABC – A – 0.02, B – 0.01, C – 0.04. Thus, the random generator will generate error with probability 0.02 (Task A). If an error occurs then process instance will stop and it will end with error. If there was an error in historic data then fitness will be increased by 1, otherwise by zero. Vice versa, if result of random generator is successful, run of process and historic data also ends successfully, fitness will also be increased by 1. This must be repeatedly executed (at least 10-40 times, the more times, the more precise result, but more computational cost).

The result of the genetic algorithm is most probable task error rates. We did not have data for this type of problem. We included this method as another future research idea.

IX. Collecting Data

We are also trying to start discussion with specialists on measurement, because measuring devices are quite expensive. There is need to join information from measurement, data mining and manufacture planning, because there is no need to measure everything. For example Half Measurement (Section III) is good example of saving money. We will now describe what is important in measurement to build simulator:

- Measure every critical task (bottlenecks). Task that are not in any critical path, don not have to be measured.
- It is good to have information about error in data, because it is problem to get it from them using only time information (Section III).
- Use Half Measurement everywhere where it is possible. Be careful, because only workplaces with low variation of production rate and high utilization are candidates for Half Measurement.
- Information about breaks is important, because we do not want to include them in production time.
- Sometimes, removable measure devices should be used to measure more tasks in different times – but be assured that nothing important has change since last measurement. Information must be valid, not obsolete.
- Machine with constant production speed, which are also independent on product attributes do not have to be measured. But be careful, sometimes preparation of product for the constant machine is dependent on product attributes, in that case, measure preparation instead of production.

X. Conclusion

In the paper, it has been shown that the quality of results does not depend only on our methods, but mainly on manufactory itself. For example, if execution time cannot be predicted from case attributes in expected precision, prediction will be useless. But this does not mean that the whole task is not predictable. Some tasks has little variance itself, so no advanced methods are needed.

In our company, predictions helped lower execution time variance, which is very useful in internal logistics planning, but there is a question what precision is needed to implement some better planning techniques that will enable significant saving especially in space and time needed for manufacturing production by improving input data for planning algorithms. We can also find a subset of case parameters that have low time deviation and try to optimize their production. Other cases could be produced in another time or in other machines in parallel with another approach (slower but more robust).

Resources working speed was also the big issue. In addition, dynamic aspect of process (new machines, resource improvement) is a problem to solve. We also tried other approaches like Association Rules and use them with success, but still, some problems are awaiting us, mainly because of unmeasured or partly measured processes. We introduced some solutions that worked and we hope that other problems will be solved too. This could be topic of our further research. We believe these methods could reach maturity and will be used in some manufactories in future.

ACKNOWLEDGMENT

This research was supported by the grants of MPO Czech Republic TIP FR-TI3 039, the grant FIT-S-10-2, the research plan no. MSM0021630528 and the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070).

REFERENCES

[1] M. Pospíšil, V. Mates, and T. Hruška, “Process Mining in Manufacturing Company,” in The Fifth International Conference on Information, Process, and Knowledge Management, Nice, France, IARIA, 2013, pp. 143-148, ISBN 978-1-61208-254-7

[2] W.M.P. van der Aalst, H.A. Reijers, A.J.M.M. Weijters, B.F. van Dongen, A.K. Alves de Medeiros, M. Song, and H.M.W. Verbeek, “Business process mining: An industrial application,” Information Systems, Volume 32, Issue 5, July 2007, pp. 713-732, ISSN 0306-4379, DOI: 10.1016/j.is.2006.05.003.

[3] M Song and W.M.P. van der Aalst, “Towards comprehensive support for organizational mining,” Decision Support Systems, Volume 46, Issue 1, December 2008, pp. 300-317, ISSN 0167-9236, DOI: 10.1016/j.dss.2008.07.002.

[4] W. M. P. van der Aalst, and A. J. M. M. Weijters, “Process mining: a research agenda”, Computers in Industry, Volume 53, Issue 3, Process / Workflow Mining, April 2004, pp. 231-244, ISSN 0166-3615, DOI: 10.1016/j.compind.2003.10.001.

[5] W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters, “Workflow mining: A survey of issues and approaches,” Data & Knowledge Engineering, Volume 47, Issue 2, November 2003, pp. 237-267, ISSN 0169-023X, DOI: 10.1016/S0169-023X(03)00066-1.

[6] W. M. P. van der Aalst, “Process Mining,” Berlin, Heidelberg 2011, ISBN 978-3-642-19344-6

[7] A. Rozinat, R.S. Mans, M. Song, and W.M.P. van der Aalst, “Discovering simulation models,” Information Systems, Volume 34, Issue 3, May 2009, pp. 305-327, ISSN 030

[8] A. Rozinat, M.T. Wynn, W.M.P. van der Aalst, A.H.M. ter Hofstede, and C.J. Fidge, “Workflow simulation for operational decision support”, Data & Knowledge Engineering, Volume 68, Issue 9, Sixth International Conference on Business Process Management (BPM 2008) - Five selected and extended papers, September 2009, pp. 834-850, ISSN 0169-023X, DOI: 10.1016/j.datak.2009.02.014.

[9] W.M.P. van der Aalst, M.H. Schonenberg, and M. Song, “Time prediction based on process mining”, Information Systems, Volume 36, Issue 2, Special Issue: Semantic Integration of Data, Multimedia, and Services, April 2011, pp. 450-475, ISSN 0306-4379, DOI: 10.1016/j.is.2010.09.001.

[10] M. Pospisil., T. Hruška, “Business Process Simulation for Predictions,” in BUSTECH 2012: The Second International Conference on Business Intelligence and Technology, Nice, France, IARIA, 2012, pp. 14-18, ISBN 978-1-61208-223-3

[11] J. Nakatumba, A. Rozinat, and N. Russell, “Business Process Simulation: How to get it right,” Springer-Verlag, 2010, doi=10.1.1.151.834

[12] J. Nakatumba and W.M.P.V.D. Aalst, “Analyzing Resource Behavior Using Process Mining”, in Proc. Business Process Management Workshops, 2009, pp. 69-80.

[13] M. Pospíšil, V. Mates, T. Hruška, “Analysing Resource Performance and its Application in Company,” in The Fifth International Conference on Information, Process, and Knowledge Management, Nice, France, IARIA, 2013, pp. 149-154, ISBN 978-1-61208-254-7

[14] W.M.P. Van der Aalst, “Business Process Simulation Revisited,” 2010, ISSN: 1865-1348

[15] B. Wetzstein, P. Leitner, F. Rosenberg, I. Brandic, S. Dustdar, and F. Leymann, “Monitoring and Analyzing Influential Factors of Business Process Performance,” Enterprise Distributed Object Computing Conference, 2009. EDOC '09. IEEE International, pp. 141-150, 1-4 Sept. 2009, doi: 10.1109/EDOC.2009.18

[16] D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, and M.C. Shan, “Business Process Intelligence, Computers,” in Industry, Volume 53, Issue 3, Process / Workflow Mining, April 2004, pp. 321-343, ISSN 0166-3615, DOI: 10.1016/j.compind.2003.10.007.

[17] D. Grigori, F. Casati, U. Dayal, and M.C. Shan, “Improving Business Process Quality through Exception Understanding,Prediction, and Prevention,” in Proceedings of the 27th VLDB Conference,Roma, Italy, 2001, 1-55860-804-4

[18] R. Agrawal, T. Imielinski., A. Swami, “Mining Association Rules Between Sets of Items in Large Databases”, Proceedings of the ACM SIGMOD Conference on Management of Data, Washington, USA, 1993, pp. 207-216.

[19] R. Agrawal, R. Srikant, “Fast Algorithms for Mining Association Rules in Large Databases,” in VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases, San Francisco, USA, 1994, pp. 487—499.

[20] J. Han, J, J. Pei, Y. Yin, “Mining Frequent Patterns without Candidate,” Proceedings of the ACM-SIGMOD Conference on Management of Data (SIGMOD'00), Dallas, TX, 2000, pp. 1-12.

[21] B. Liu, W. Hsu, and Y. Ma, “Integrating Classification and Association Rule Mining,” in ACM Conference on Knowledge Discovery and Data Mining (SIGKDD’98), New York, August 1998, pp. 80–86.

[22] W. Li, J. Han, and J. Pei.: CMAR, “Accurate and efficient classification based on multiple class-association rules,” in IEEE International Conference on Data Mining (ICDM’01), San Jose, California, 2001, pp. 369 – 376.

[23] M. L. Antonie, and O. Zaiane, “Text Document Categorization by Term Association,” in Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02), Maebashi City, Japan, 2002, pp. 19-26.

[24] V. Bartík, “Association Based Classification for Relational Data and Its Use in Web Mining,” in IEEE Symposium on Computational Intelligence and Data Mining, Nashville, USA, 2009, pp. 252-258.