Predictions in information systems
a process mining perspective

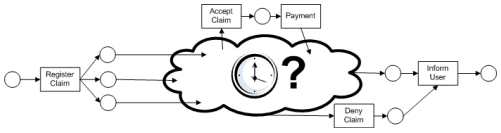
Crooy, R.A.

Award date:
2008

Technische Universiteit Eindhoven
Department of Mathematics and Computer Science

Predictions in Information Systems

a process mining perspective



Ronald Crooy, R.A.Crooy@gmail.com

November 10, 2008

Master's thesis

Supervisors: prof.dr.ir. Wil van der Aalst
dr.ir. Boudewijn van Dongen
Referee: dr.ir. Hajo Reijers

for my anonymous reviewer

Abstract

Most part of our lives we have to wait for something or someone, and although waiting is inevitable, it would be a big improvement if we knew how long we needed to wait. Most commonly we wait for some process to end, for example for the payment of an insurance claim. Not only do we always have to wait for the process to end, we are also eager to know the outcome of the process. If our insurance company is able to tell us in advance that it is very likely that our claim is be accepted in 2-3 weeks, it could make our waiting less of a burden. The organizations we deal with are supported by information systems, which store a great deal of information. Using all of this information it is possible to give a better prediction on e.g. our remaining waiting time. Many obstacles exist in giving such a better prediction, such as privacy laws or the size and complexity of a process.

In this thesis we describe a prediction method developed in the context of process mining.This prediction method uses non-parametric regression in order to make predictions on the total remaining cycle time, the remaining cycle time until a certain activity and the number of occurrences of an activity. The prediction method does this with as little assumptions as possible, using a mix of continuous and (un)ordered discrete data. As is customary in process mining, the prediction method uses an event-log to derive all information needed for the predictions. The prediction method described has been implemented as a plugin for ProM, a process mining framework.

In order to validate the method, a case study was performed using a case taken from practice. The case study shows that the prediction method we describe performs better than the average based predictor, which is a very simple predictor that bases a prediction on the average remaining cycle time. Furthermore, the prediction method was tested in a number of simulations. The results of these simulations give some insights into the prediction method and shows that the prediction method performs similar or better than the average based predictor, given that the prediction method can use the best set of predictor variables.

In conclusion, we found that the non-parametric prediction has a good performance, and is easy to apply to any process without pre-existing knowledge on the underlying process.

Contents

1 Introduction

1.1 The problem

"A sure way to irritate people and to put evil thoughts into their heads is to keep them waiting a long time. This makes them immoral."
-Friedrich Nietzsche (1844 - 1900).

Most part of our lives we have to wait for something or someone, and although waiting is inevitable, it would be a big improvement if we knew how long we needed to wait. This is not as easy as one would imagine. Most commonly we wait for some process to end, for example for the payment of an insurance claim. Not only will we always have to wait for the process to end, we are also eager to know the outcome of the process. If our insurance company could tell us in advance that it is very likely that our claim will be accepted in 2-3 weeks, it could make our waiting less of a burden.

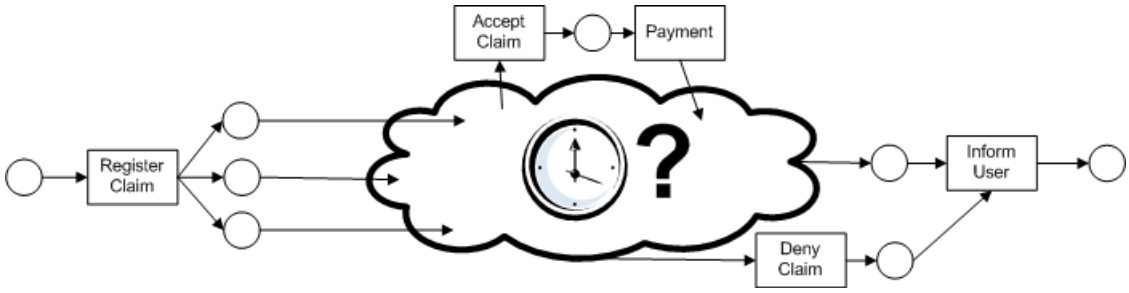


Figure 1: Imaginary insurance claim process

Imagine a simplified insurance claim handling process as shown in Figure 1. After an insurance claim is filed, a process begins that eventually results either in a payment or not. For example, the claim is checked to verify that it is filled in correctly, it is checked for possible fraud, etc. Depending on the outcome of each activity, different activities follow that influence the waiting time. There was a time when an employee of your local insurance office was able to know the entire process and knew enough about the customer to give a pretty good estimate of the remaining waiting time. As times changed, so did these processes and the technology supporting them. Nowadays, information systems have automated administrative processes and computers have taken over many jobs from humans. The processes have grown more complex, are typically distributed over multiple organizations, and are less transparent, making it impossible for an employee to understand the complete system. In fact, an employee does not need to know the details of every customer or case in the process. In many cases that would even be undesirable because of privacy laws.

Fortunately, the information systems that support administrative processes store all the information they process and can use this to give an estimate for the remaining waiting

time. The time we wait, i.e. the time remaining until some event, is usually predicted to be the time you have spent so far subtracted from the average total duration. For example, if the average customer has to wait 3 to 5 weeks and you have been waiting for 1, on average you still have to wait 2 to 4 weeks. This answer is based on the average (4 weeks) plus/minus a confidence interval (1 week) to create a prediction (3 to 5 weeks) suited for usually 90% to 95% of the cases. The prediction whether or not a specific activity will occur (e.g. payment or no payment in the insurance example) can also be calculated by taking the average probability of payment in any insurance claim.

However, the customer that files the insurance claim might have committed fraud in the past, in which case the insurance claim is more likely to be thoroughly investigated. The 3 to 5 weeks are probably not a good estimate for the remaining waiting time in this case, and the likelihood of payment becomes very low. Similarly, if by some irregularity an insurance claim of 1 week old got stuck in the process for 5 weeks, then a predictor based on the average predicts that this case is almost finished. In reality the remaining waiting time has not changed during the 5 weeks, which could be predicted as nothing in this case has changed. Although predicting the remaining cycle time is between x and y days ensures a statistically correct prediction, in many cases you like to have a more precise estimate. In a hospital for example; each day a patient is in the hospital, he is costing his insurance company money and a replacement must be found at his work. Having a more accurate estimation (e.g. 3 weeks and 2 to 4 days) would make it easier for his boss to find a replacement, and would provide a more accurate cost assessment for the insurance company. Employees are not able to personalise the predictions, as this would be expensive, inefficient and impractical. In this thesis we show that, using a software tool, we are able to give a customer a more personalised prediction based on information already known. At the start, when little is known about the case at hand, the average prediction is a very good prediction. However as more information becomes available, a substantial better prediction can be given.

1.2 Requirements for a prediction method

In order to provide criteria for the selection of a suited prediction method, we formulate a set of requirements that a possible prediction method should conform to. A prediction method :

- must give accurate predictions.
- must be easy to use and give on-demand predictions.
- can only use what is in the event-log.
- must use all *types* of information contained in the event-log.
- must be able to distinguish relevant from irrelevant information.

- must make prediction with as few assumptions on the underlying process as possible.

Given this list of requirements we can compare the different methods, and select the method that conforms best to our requirements. In the following section we describe the approach we have chosen in general.

1.3 Approach

The data that information systems store can be used to construct an *event-log*, which is a sequential list detailing all activities executed concerning a specific case. Because of the availability of such a detailed event-log, it is possible to give a better (more personalised) prediction. The event-log contains information such as the sequence of activities that have occurred during a case, their durations, and possible data-attributes[1] of cases or activities. The prediction method we present uses all of the information available for its predictions, and is able to filter relevant information from irrelevant information. In order to make predictions with as little assumptions as possible, we use non-parametric regression [17]. Non-parametric regression is a form of regression that bases a regression estimate directly on the known data, instead of creating a parameterized model that computes an estimate.

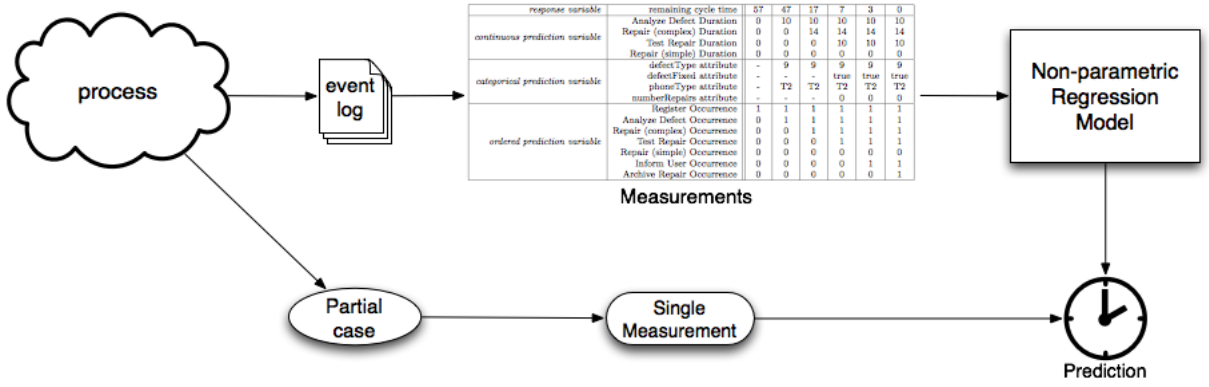


Figure 2: Overview of the prediction method

Figure 2 shows an overview of the prediction method. As mentioned, the prediction method uses an event-log. The event-log contains information corresponding to a set of historical cases. For each case we construct *measurements*, these measurements consist of a table with the *predictor variables* and a *response variable*. Predictor variables are the variables on which we base a prediction (e.g. the duration of an activity), and a response variable is the value which we would like to predict (e.g. the remaining cycle time). Using the measurements obtained from the event-log we create the non-parametric regression

[1] e.g. the person responsible for a case, whether or not a person has committed fraud, etc.

model, this part of the prediction method is (algorithmically speaking) the most complex. However, once the non-parametric regression model is created, making a prediction is a straightforward process.

First, the partial case for which we need a prediction, is converted to a measurement. Then all measurements of the historical cases are compared to the single measurement of the partial case, each historical measurement receives a weight that indicates the relative similarity of the historical case to the partial case. The predicted value is the weighted average over the response variables of all cases.

1.4 Non-parametric regression

In the previous section we already described the prediction method in general, and mentioned the use of *non-parametric regression* [17]. Non-parametric regression is a less-known form of regression that, unlike parametric regression, bases an estimation on a weighted average of predefined measurements. While parametric regression computes a response value directly on the values of predictor variables of the new case, non-parametric regression compares the new case to the set of historical cases and computes a response value on a weighted average of the most similar (old) cases. As a result, the non-parametric regression requires no information on the underlying probability distributions, or the relations between predictor variables.

Non-parametric regression is closely related to *case-based reasoning* [13]. In case-based reasoning, a new case is compared to a set of historical cases, after which the most similar cases are selected. This subset of similar cases are then used to formulate new prediction. Non-parametric regression is very similar, however it uses a more statistical approach. *By using non-parametric regression, the prediction method requires no assumptions or existing knowledge on the underlying process, and is able to base predictions solely on the information contained in the event-log.*

As mentioned in the previous section, creating the non-parametric regression model is (algorithmically speaking) the most complex part of the prediction method. In order to create the regression model, the various predictor variables in the measurements must be tested for their relevance in the comparison of cases. A "bandwidth" is used to for each predictor variable to scale the weights and possibly smooth out or increase the effect of a predictor variable. The regression model therefore determines the bandwidths for each predictor variable. The bandwidths determine which predictor variables are relevant. With the (old) measurements, and the bandwidth, the regression model is able to return a weight that indicates the similarity of a historical case to a partial case. Using this weight for each case, a local (weighted) average on the response variables is created that is the prediction for the new case.

1.5 Process Mining

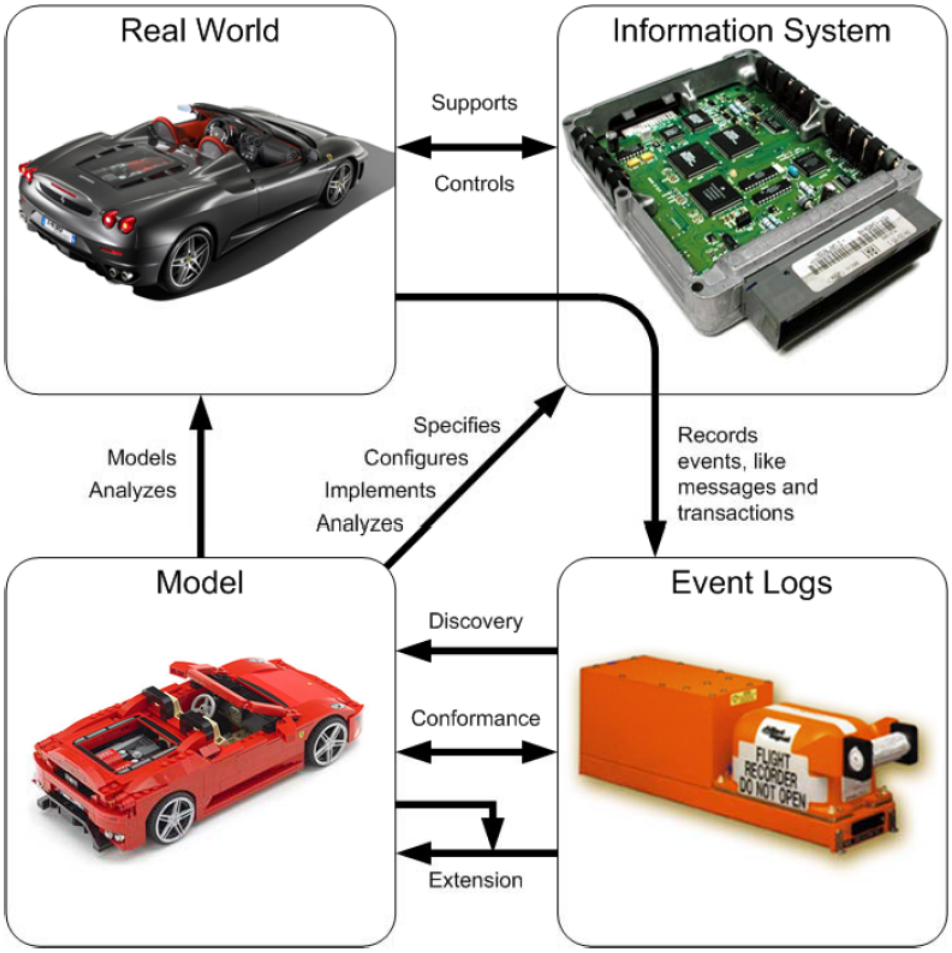


Figure 3: Process Mining, picture taken from [35].

The process of extracting information from the event-log to enrich the information system containing that process, is part of a technique called *process-mining* [3, 2, 5]. Process-mining enables the extraction of information from event-logs, such as the discovery of (process) models that describe a process that is supported by an information system. The technique is related to data-mining, however, process-mining focusses on dynamic behaviour. With the help of process-mining it is possible to discover, monitor, and improve real processes, instead of assumed processes.

The prediction method discussed in this thesis is created and described in the context of process-mining. As in process-mining, the prediction method described in this thesis uses an event-log to create a prediction model that models the real world. The prediction model in turn, is used by an information system to support the real world with predictions of

e.g. the remaining cycle time. Figure 3 illustrates the role of Process Mining with respect to information systems. The prediction method presented in this paper is best illustrated in this model as a combination of both "Model" and "Information System", as the prediction method uses an event-log to model the "Real World", but is mostly used to support the "Real World" with predictions.

1.6 ProM/Process Mining Group

The Process Mining Group[2] at the Eindhoven University of Technology is a cooperation between the department of Mathematics and Computer Science and the department of Technology Management. This group is responsible for the development of the ProM Framework [14, 1]. ProM is a highly extensible framework that supports various process mining techniques by the use of plugins, it is capable of importing from and exporting to various information systems, workflow engines, and classical mining tools. ProM is developed in the platform-independent programming language Java, and is currently licenced under Common Public Licence (CPL). The topic of this thesis was proposed in close cooperation with the Process Mining Group. In fact, the prediction method described is implemented as a plugin for the ProM Framework.

1.7 Outline

This thesis is structured in the following way; in the next chapter we discuss relevant work, more specifically we discuss various approaches and show that the approach chosen is reasonable given the requirements for a prediction method. In Chapter 3 non-parametric regression is explained in detail. An example is used throughout this chapter to make the abstract concepts behind the method more concrete. After the explanation of non-parametric regression, Chapter 4 describes how non-parametric regression is applied to predictions in a process-mining context. The same example as given in Chapter 3 is used to clarify certain methods. To validate our approach, Chapter 5 shows a number of simulations that illustrate important aspects of the prediction method. Also this chapter contains a case-study that shows how the prediction method performs in a case taken from practice. The ProM plugin that implements the prediction method we discuss, is described in Chapter 6. In Chapter 7 we present some proposals for future work, in order to stimulate researchers to continue researching this prediction method. Finally we reflect on the results and draw conclusions in Chapter 8.

[2] http://www.processmining.org

2 Related Work

The problem of making predictions based on event-logs is summarised in two main points. One is that the log contains different types of data, and the other is that there is no prior knowledge available except what is in the log. With respect to the data in the log, we distinguish three types:

- continuous data, such as the duration of an activity,
- discrete ordinal data (or ordered), such as the number of times an activity occurred,
- discrete nominal (or categorical) data, such as the person responsible for an activity, or the type of insurance policy,

With respect to the prior knowledge, we assume that the only known information is contained in the log. Things like the activities that occurred, the time they took, people who executed them, etc. The goal is to make accurate predictions with as little assumptions on the process, or any aspect of the process, as possible. In this chapter we first discuss some problem related work, and we discuss different methods and their suitability. In conclusion we motivate the chosen approach.

2.1 Problem related work

"2G Case Prediction in Staffware" The TIBCO Staffware iProcess Suite [32] is one of the leading Business Process Management Suites (BPMS) on the market, and one of the few that support case prediction. The prediction method used in Staffware, however, is identified to have some shortcomings [29]. More specifically, the prediction method assumes that each case takes the shortest path through the process, where all activities are predicted to have constant cycle times. In other words, the prediction method in Staffware is very optimistic, and assumes the cycle times of activities do not vary.

"Case Prediction in BPM systems: A Research Challenge" The challenges in case prediction for Business Process Management are described in [27]. In short, any prediction method presented for case prediction must be accurate, easy to use, fast (on-demand predictions) and it should not interfere with the operation of the BPMS. Similar to [29] this paper also mentions the shortcomings in the Staffware prediction method.

"Case prediction in BPM systems: Research to the predictability of the remaining time of individual cases" A study into the predictability of remaining cycle time, is shown in [12]. This master's thesis shows that research into predicting cycle times

in BPM systems is mostly focussed on project management and scheduling, instead of providing accurate predictions for customers or clients. Furthermore, this work shows in an experiment that it is possible to provide better prediction than what is normally accepted.

"Cycle Time Prediction: When Will This Case Finally Be Finished?" The work presented in this thesis is also presented in [15], this paper is a result of the work done for this thesis and therefore relevant work. This paper has been written in parallel with this thesis, and is to be presented on the COOPIS 2008 conference.

2.2 Related work using simulations for predictions

"Workflow Simulation for Operational Decision Support using YAWL and ProM" Simulations are normally used for analysing the steady-state behaviour of processes. However, [28] describes how simulations can be used for operational decision support. By combining simulation with techniques from Process mining [14], it is possible to make predictions for a single case. A simulation model is created using the designs of the information system, next by using historical information (retrieved by process mining techniques) the cycle time distributions of activities are determined, and finally the information system provides the simulation with current-state information. This method makes use of the event-log of information systems, which is precisely what is available in the context of the problem in this thesis. However, because the designs of an information system are not always available or up to date, for this thesis we assume that this information is not present. In conclusion, this method requires more information than is available, and is therefore not a feasible method.

2.3 Related work using stochastics for predictions

"Estimating cycle times in stochastic Petri nets" One approach, as described in [6], estimates the cycle times by using Petri nets and methods from stochastic processes under the assumption that all process times are independent and have an equal time probability distribution. Although, with the use of a technique called process mining [3] it is possible to create a Petri net of the process, these Petri nets are not always correct, as domain knowledge may be required to retrieve the correct Petri nets.

Furthermore the method described in [6] requires the probability distribution of process times of activities to be independant and equal. Because we have no knowledge on the probability distribution of the process times of activities, we make assumptions about this probability distribution. Also it has to be assumed that this probability distribution is equal for all activities, and that the process times of all activities are independent. Although we are able to retrieve the process times of the activities from the log, the information on the

process times is not sufficient to determine their probability distributions, or if they are all equal. In conclusion, the method described in [6] requires more information than available in the log, also this method only uses a fraction of information that is typically available, e.g. this method does not consider the process times of past activities or data-attributes. Therefore, this method is not conforming to our requirements.

2.4 Related work using multiple linear regression for predictions

"Cycle time estimation for printed circuit board assemblies" A method proposed in [16], shows how cycle times can be predicted for the manufacturing of printed circuit boards using multiple linear regression. A linear parameterized model that is used to make predictions is made using multiple linear regression. Therefore it must be assumed that the process that generated the log can be modelled by a linear parameterized model. Although this method does use all the available data from the log, it also requires making assumptions on the underlying process. It is therefore also not a feasible method.

2.5 Predictions using a combination of techniques

"Factory cycle-time prediction with a data-mining approach" Using a combination of clustering and regression trees, the method described in [7] predicts the cycle time for batches of products in semi-conductor manufacturers. The data is first analysed for clusters based on the type of case after which a regression tree is created for each cluster. The method proposed in this paper is in theory also applicable to our problem. However, this method is meant for a specific type of process, namely the process of manufacturing semi-conductors. A semi-conductor manufacturer is highly automated, meaning there is little variance in the speed of cases. Therefore, a correct assumption in a semi-conductor manufacturing process is that a new batch with equal properties as another batch should have an (almost) equal cycle time. Whereas in an information system for the average administrative process, every case is different, because the mainly human resources are not as constant in production as machines in a semi-conductor manufacturer. Therefore, two cases with equal properties can result in two completely different cycle times. This type of variance in the measurements means the use of the classification resulting from a regression tree is not the best choice for a prediction model.

In summary, this method is able to use all data in the log, based solely on what is available from the log. But this method does not appear to be suited for information systems, as it was designed for a semi-conductor manufacturer with more predictable resources and less variation in the cycle times of activities.

"Cycle time prediction and control based on production line status and manufacturing data mining" Comparatively, the work in [9] shows a combination of techniques such as clustering, regression trees, polynomial regression and sensitivity analysis, and also focusses on the cycle times of batches in a semi-conductors manufacturing plant. The method is effectively not that different from the method chosen in [7], but consists of more steps, specifically sensitivity analysis and polynomial regression. This method does not provide benefits over the previously described method, it does however add more complexity to the prediction method itself. Therefore it is not suited for the same reasons as the previously described method.

2.6 Predictions using nonparametric regression

"A statistical approach to case based reasoning, with application to breast cancer data" In Section 1.4 we introduced non-parametric regression, [13] shows an application of non-parametric regression on a large dataset of breast cancer patients with no knowledge on the aspects of the information in the dataset. This problem is comparable, as it describes the prediction of survival times, which are a specific form of cycle times. Although the title of [13] refers to case-based reasoning, an adapted form of non-parametric regression is used that is best described as a more statistical form of case-based reasoning.

In summary, the prediction method as it is described in [13] and [17] requires no assumptions on the underlying process, however, this applied form of non-parametric regression uses only continuous data as a bases for predictions.

"Nonparametric estimation of regression functions with both categorical and continuous data" The non-parametric regression described in [25] is the first form of non-parametric regression that combines continuous and discrete (un)ordered data in a single model. Therefore, a prediction method using the non-parametric regression as described in [25] is capable of making predictions with no assumptions or preexisting knowledge on the underlying data or processes, and is capable of using all types of data as a bases for the predictions.

2.7 Approach

Of all approaches described, both the method using a combination of clustering and regression trees (Section 2.5) and the non-parametric regression (Section 2.6) conform to our requirements (Section 1.2). The prediction method using combination of regression trees and clustering is, compared to non-parametric regression, less suited for the prediction of cases with few to no precedents, because regression trees are usually used for classifications and not for the prediction of continuous information (like the remaining cycle time). *Non-*

parametric regression is therefore the method we use in the prediction method described in this paper. In the following chapter we describe non-parametric regression in detail.

3 Non-parametric regression

3.1 The running example

As previously mentioned, the prediction method described in this thesis relies on non-parametric regression. In this chapter we explain non-parametric regression in detail. In order to make certain abstract concepts more concrete, we use a running example. The running example is based on a log of a simulated process, that describes how repairs are made to the houses of a housing rental organisation. The log of this process is found in [11]. The log contains 500 cases, with 4 different data-attributes ("phoneType", "defectType",

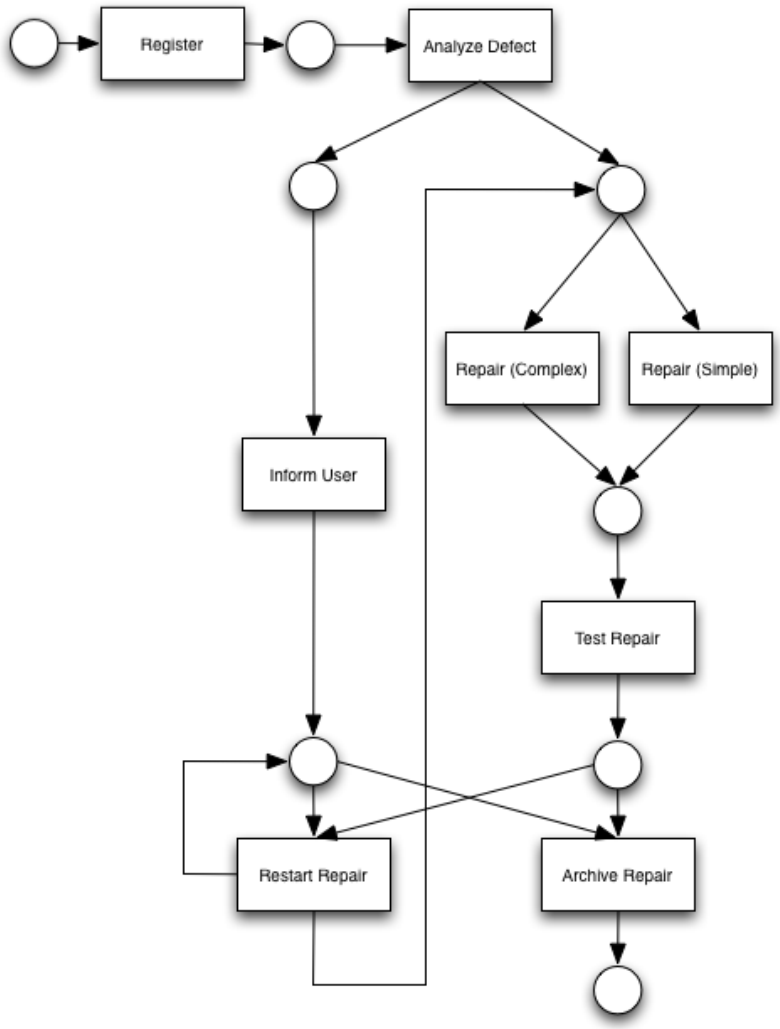


Figure 4: Petri net of the running example

"defectFixed" and "numberRepairs"), 8 different activities ("Analyze Defect", "Archive Repair", "Inform User", "Register", "Repair(complex)", "Repair(simple)", "Restart Repair" and "Test Repair") of which 4 activities ("Analyze Defect", "Repair(complex)", "Repair(simple)", and "Test Repair") have recorded duration's. Also the process contains a loop, meaning some activities may occur multiple times. The process, shown in Figure 4, starts by registering a defect, which is then analyzed. Finally a loop starts that repairs (simple or complex) and tests the defect until it is repaired.

We have to stress that the process model shown in Figure 4 is constructed using a plug-in of ProM [14]. This process model is not available to our regression model and is provided only for reference purposes.

In Section 1.4, we described the non-parametric prediction method in general. We described that an information system collects data concerning a certain process, and that this data is used to construct an event-log. The event-log is used to derive the measurements that are used to create a non-parametric regression model. Because we focus on non-parametric regression in this chapter, we do not describe how the measurements are derived from the event-log. Instead we use an example with simplified measurements to explain non-parametric regression. The derivation of measurements from a log, their application in the non-parametric regression are described in detail in Chapter 4.

Case properties				
predictor variables	*case 1*	*case 2*	*case 3*	*case 4*
Analyze Defect Duration(x_1)	10	5	9	7
Repair (complex) Duration(x_2)	14	44	10	0
Test Repair Duration (x_3)	10	7	9	9
Repair (simple) Duration (x_4)	0	0	0	5
Analyze Defect Occurrence (x_5)	1	1	1	1
Repair (complex) Occurrence(x_6)	1	1	1	0
Test Repair Occurrence (x_7)	1	1	1	1
Repair (simple) Occurrence (x_8)	0	0	0	1
phoneType Attribute(x_9)	T2	T2	T1	T1
defectType Attribute(x_{10})	9	8	6	1
defectFixed Attribute(x_{12})	true	true	true	true
numberRepairs Attribute(x_{12})	0	0	0	1
Abstract Response Variable	Y_1	Y_2	Y_3	Y_4

Table 1: Simplified measurements of four cases of the running example

Table 1 shows the simplified measurements of four cases taken from the simulated process in Figure 4. These four cases are shown in full in Appendix G on page xix. In this example we intentionally leave the response variables abstract, as these are not relevant for the purpose of the running example at this point. The running example is described in more detail in Chapter 4, at which point we use the running example to illustrate the derivation of measurements. As described in Chapter 2, we distinguish three types of data; continuous data, discrete ordinal (or ordered) data, and discrete nominal (or categorical)

data. An example of continuous data in Table 1 is the duration (in minutes) of an activity and although they happen to be natural numbers, they should be seen as real numbers. An example of discrete ordinal data is the number of occurrences of an activity. In Table 1 the occurrences of activities are all either 0 or 1, however these are natural numbers and not limited to 0 or 1. The last type, the discrete nominal data is shown in Table 1 in the form of data-attributes. Discrete ordinal values are best described as categories without ordering, categories are either equal or not equal. Also, categories can never be put in any order, as opposed to e.g. natural numbers, which are discrete ordinal (or ordered) data. Note that, although we assume all data-attributes are categorical, they are not. For example, the data-attribute "phoneType" is clearly categorical, while the numerical data-attribute "numberRepairs" is very likely a discrete ordered variable. Although it judging by "numberRepairs" and "phoneType" it appears that we can distinguish all data-attributes with numerical values as discrete ordered and all non-numerical values as categorical, we can not. A good example of where this goes wrong is with the data-attribute "defectType", which appears to be an ordered discrete variable judging by its numerical value, whereas its name would indicate it is categorical. This shows that it is impossible to distinguish whether a data-attribute is ordered or categorical, therefore we must assume that all data-attributes are categorical.

The goal of regression analysis is to create a regression model on which we can accurately estimate an unknown response variable based on a set of measurements. These measurements are pairs of the predictor variables and the corresponding actual response variables. In our case the predictor variables are the data contained in the log, such as the occurrences of activities, the durations of those activities and possible data-attributes. The response variables are the total remaining cycle time, the remaining cycle time until activity a, or the occurrence of a. In the remainder of this chapter we describe non-parametric regression, assuming that we already have measurements and we use the example introduced here to illustrate the math involved. Note that Chapter 4 describes how the measurements are obtained.

3.2 Preliminaries

Let $p \in \mathbb{N}$ and $\mathcal{C}$ an unspecified abstract domain, we define $\vec{x} \in \mathcal{C}^p$ to be a p-dimensional *vector* of $\mathcal{C}$, also expressed by $\vec{x} = (x_1,\ x_2,\ x_3,\ \ldots, x_p)$. For a vector $\vec{x} \in \mathcal{C}^p$, x_i denotes the i^{th} component of that vector. We define *scalar multiplication*, i.e. the product of a vector $\vec{x}$ and a number a, by $a\vec{x} = (ax_1,\ ax_2,\ ax_3,\ \ldots\ , ax_p)$.

3.3 The basic idea

Given a set of measurements $\{(Z_i, \vec{X}_i, Y_i)\}_{i=1}^{N}$ where Z_i is an unique identifier for each tuple, $\vec{X}_i = (X_{i,1}, X_{i,2}, \ldots, X_{i,p})$ is the vector with predictor variables on some abstract

domain $\mathcal{C}^p$ and $Y_i \in \mathbb{R}$ is the corresponding response variable. The unique identifier Z_i is needed to allow for duplicate measurements in the set. At this point we do not define Z_i in more detail, however in Chapter 4 we show how to construct a unique identifier. We define $\gamma : \mathcal{C}^p \to \mathbb{R}$ to be a regression function that models the relationship between Y_i and $\vec{X}_i$ in Equation (1). As we estimate the response variable Y_i as good as possible we try to minimize the error ϵ_i.

$$Y_i = \gamma(\vec{X}_i) + \epsilon_i \qquad , \epsilon_i \mapsto 0 \tag{1}$$

Non-parametric regression is based on a technique called *local averaging* (also known as smoothing) in which the estimate y' for a y of a measurement $\vec{x} = (x_1, x_2, \ldots, x_p)$ is based on the known cases $\{(Z_i, \vec{X}_i, Y_i)\}_{i=1}^N$. The local average is constructed such that the estimate y' is a weighted average of the cases from $\{(Z_i, \vec{X}_i, Y_i)\}_{i=1}^N$ that are "closest" to $\vec{x}$. We define $\gamma(\vec{x})$ as the smoothing function in Equation (2).

$$\gamma(\vec{x}) = \frac{\sum_{i=1}^N \delta(\vec{X}_i, \vec{x}) \cdot Y_i}{\sum_{i=1}^N \delta(\vec{X}_i, \vec{x})} \tag{2}$$

In this equation $\delta : (\mathcal{C}^p \times \mathcal{C}^p) \to \mathbb{R}$ is a weight-function that returns a weight corresponding to the relative distance between $\vec{X}_i$ and $\vec{x}$. The higher the weight returned by δ, the closer and more important $\vec{X}_i$ is for $\vec{x}$. Note that the divisor of Equation (2), $\sum_{i=1}^N \delta(\vec{X}_i, \vec{x})$ is the sum of relative distances between $\vec{x}$ and each point $\vec{X}_i$, or the density of $\vec{X}_i$. As the weighted sum over Y_i is divided by the density, a low density (meaning few cases are close to $\vec{x}$) results in a bigger influence for the cases closest to $\vec{x}$. Equally a higher density (meaning more cases are close to $\vec{x}$) results in a smaller influence for each of the cases close to $\vec{x}$. In this way the local average is obtained.

The example we presented in Table 1 in the beginning of this chapter contains all three types of data, the durations of activities are continuous variables (i.e. minutes, hours, etc.), the number of occurrences of each activity is a natural number (i.e. a discrete ordered variable, such as the number of times a repair occurred), and the data-attributes are assumed to be categorical (i.e. discrete unordered variables). Each of these three types of predictor variables are contained within the vectors $\vec{X}_i$ and $\vec{x}$. So far we have associated $\vec{X}_i$ and $\vec{x}$ to some abstract domain $\mathcal{C}$. However, $\vec{X}_i$ and $\vec{x}$ are vectors containing values in $\mathbb{R}_0^+$, $\mathcal{D}_u$ and $\mathcal{D}_o$, in which $\mathcal{D}_u$ is the domain of the categorical predictor variables, and $\mathcal{D}_o$ is the domain of the discrete ordered predictor variables. Both discrete domains $\mathcal{D}_u$ and $\mathcal{D}_o$ are subsets of $\mathcal{D}$, the domain of all discrete predictor variables. Note that actually each discrete predictor variable has its own domain, however for practical reasons we define the discrete predictor variables to have the abstract domain $\mathcal{D}$.

$$\vec{X_i} = (\underbrace{X_{i,1}, \ldots, X_{i,m}}_{\in \mathbb{R}_0^+}, \underbrace{\underbrace{X_{i,m+1}, \ldots, X_{i,m+l}}_{\in \mathcal{D}_u}, \underbrace{X_{i,m+l+1}, \ldots, X_{i,p}}_{\in \mathcal{D}_o}}_{\in \mathcal{D}}) \tag{3}$$

We define both $\vec{X_i}$ and $\vec{x}$, as vectors of m continuous predictor variables, l categorical predictor variables, $p-(m+l)$ ordered predictor variables, and k as the number of discrete predictor variables, i.e. $k = p - m$. This is illustrated by Equation (3), which shows how $\vec{X_i}$ is structured, and formalised in Equation (4). Note that, this definition of $\vec{X_i}$ and $\vec{x}$ allows for a vector with e.g. 0 continuous predictor variables, or any other combination of one or more types of predictor variables.

Definition 3.1 (Measurements Domain)**.**
Given $m, l, p \in \mathbb{N}$, we define the predictor variables $\vec{X} = (\vec{X_1}, \vec{X_2}, \ldots, \vec{X_N})$ and $\vec{x}$ such that:

$$\left(\forall a, b, c, i \in \mathbb{N} : \begin{array}{l} 1 \leq i \leq N \wedge \\ 1 \leq a \leq m \wedge \\ m+1 \leq b \leq m+l \wedge \\ m+l+1 \leq c \leq p \end{array} : \begin{array}{l} x_a, X_{i,a} \in \mathbb{R}_0^+ \wedge \\ x_b, X_{i,b} \in \mathcal{D}_u \wedge \\ x_c, X_{i,c} \in \mathcal{D}_o \end{array} \right) \tag{4}$$

In the remainder of this chapter we discuss the different weight functions that correspond to the different predictor variables. First we discuss the continuous weight function, and then the discrete weight function.

3.4 The continuous weight function

In this section we discuss the weight function $\delta^c : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ for a continuous predictor variable. For readability we define $\vec{X_i^c} = (X_{i,1}^c, X_{i,2}^c, \ldots, X_{i,m}^c), \vec{x^c} = (x_1^c, x_2^c, \ldots, x_{i,m}^c) \in \mathbb{R}^m$ such that $(\forall j \in \mathbb{N} : 1 \leq j \leq m \leq p : X_{i,j}^c = X_{i,j} \wedge x_j^c = x_j)$, i.e. $\vec{X_i^c} = (X_{i,1}, X_{i,2}, \ldots, X_{i,m})$ and $\vec{x^c} = (x_1, x_2, \ldots, x_{i,m})$ are continuous predictor variables. The goal of δ^c is to return a weight for a measurement $\vec{X_i^c}$, based on its distance to $\vec{x^c}$. As $\vec{X_i^c}$ and $\vec{x^c}$ are both vectors of individual predictor variables, the weight function returns the product of the weights of each of the m individual dimensions of $\vec{X_i^c}$ and $\vec{x^c}$. Equation (5) shows the continuous weight function, where $\vec{h^c} \in \mathbb{R}^m \wedge (\forall j \in \mathbb{N} : 1 \leq j \leq m : h_j^c > 0)$, i.e. $\vec{h^c} = (h_1^c, h_2^c, \ldots, h_m^c)$ is a m dimensional vector of real numbers larger than 0. The so-called bandwidth $\vec{h^c}$ is used to increase or decrease the effect of a certain dimension of $\vec{X_{i,j}^c}$ on the product weight of $\vec{X_i^c}$. The use of the bandwidth variables are described later in this chapter. The vector of bandwidth variables $\vec{h^c}$ is a part of the vector $\vec{h} = (h_1, \ldots, h_p)$, such that $(\forall j \in \mathbb{N} : 1 \leq j \leq m : h_j = h_j^c)$.

$$\delta^c(\vec{X_i^c}, \vec{x^c}) = \prod_{j=1}^{m} (h_j^c)^{-1} \cdot \kappa(\frac{x_j^c - X_{i,j}^c}{h_j^c}) \tag{5}$$

The weight function δ^c makes use of a so-called *kernel function* $\kappa : \mathbb{R} \to \mathbb{R}^+$. The kernel function is a symmetrical, continuous and bounded function with $\int \kappa(u)du = 1$, that can have a number of forms which are shown later. Substitution of Equation (5) in (2) yields the smoothing function in Equation (6), also known as the *Nadaraya-Watson kernel estimator* [24, 34].

$$\gamma(\vec{x^c}) = \frac{\sum_{i=1}^{N} \prod_{j=1}^{m} (h_j^c)^{-1} \cdot \kappa(\frac{x_j^c - X_{i,j}^c}{h_j^c}) \cdot Y_i}{\sum_{i=1}^{N} \prod_{j=1}^{m} (h_j^c)^{-1} \cdot \kappa(\frac{x_j^c - X_{i,j}^c}{h_j^c})} \tag{6}$$

The Nadaraya-Watson kernel estimator in Equation (6) uses a scale factor or *bandwidth* $\vec{h^c}$ to scale the weights, whereas the kernel function κ is used to determine the shape of the weight distribution. The most simple kernel function is the Uniform kernel function $\kappa(u) = \begin{cases} \frac{1}{2} & , |u| \leq 1 \\ 0 & , |u| > 1 \end{cases}$, i.e. a function that, if combined with Equation (5), results in a weight function which returns a weight $\frac{1}{2}$ for points where the difference between x_j^c and $X_{i,j}^c$ divided by h_j^c is smaller than 1, and 0 for all other points. In other words, this is a weight function that returns a weight of $\frac{1}{2}$ to all points where the difference (or distance) between x_j^c and $X_{i,j}^c$ is smaller than h_j^c. The kernel function used in this thesis is the (second order) Gaussian kernel, as shown in Equation (7).

$$\kappa(u) = \frac{e^{-u^2/2}}{\sqrt{2\pi}} \tag{7}$$

The Gaussian kernel is a kernel that has continuous support, meaning that every point receives a weight although it may be close to 0 (but not 0). If the density of the measurements is very low (meaning there is a high variance in the measurements) and there are few to no measurements that are "close", then using a kernel with limited support (such as the Uniform kernel) results in a prediction based on very few measurements. However, a kernel with continuous support returns a non-zero weight for each point, resulting in a prediction based on all measurements (although the influence of some might be negligible). The software package NP [19], which we based our software on, provides regression functions with the Uniform kernel, the Gaussian kernel, and the Epanechnikov kernel function. The Epanechnikov kernel function is defined as $\kappa(u) = \begin{cases} \frac{3}{4}(1-u^2) & , |u| \leq 1 \\ 0 & , |u| > 1 \end{cases}$, and like the Uniform kernel it is bounded. Although many more kernels exist, these three kernels are most commonly used. Figure 5 shows the three kernels, in this plot it is visible how each kernel has an influence on the distribution of the weights. The Epanechnikov kernel

results in a much higher influence for very similar cases than the Gaussian kernel. Also it is visible how the Gaussian kernel has continuous support, whereas the other kernels only return a non-zero value for between 1 and −1.

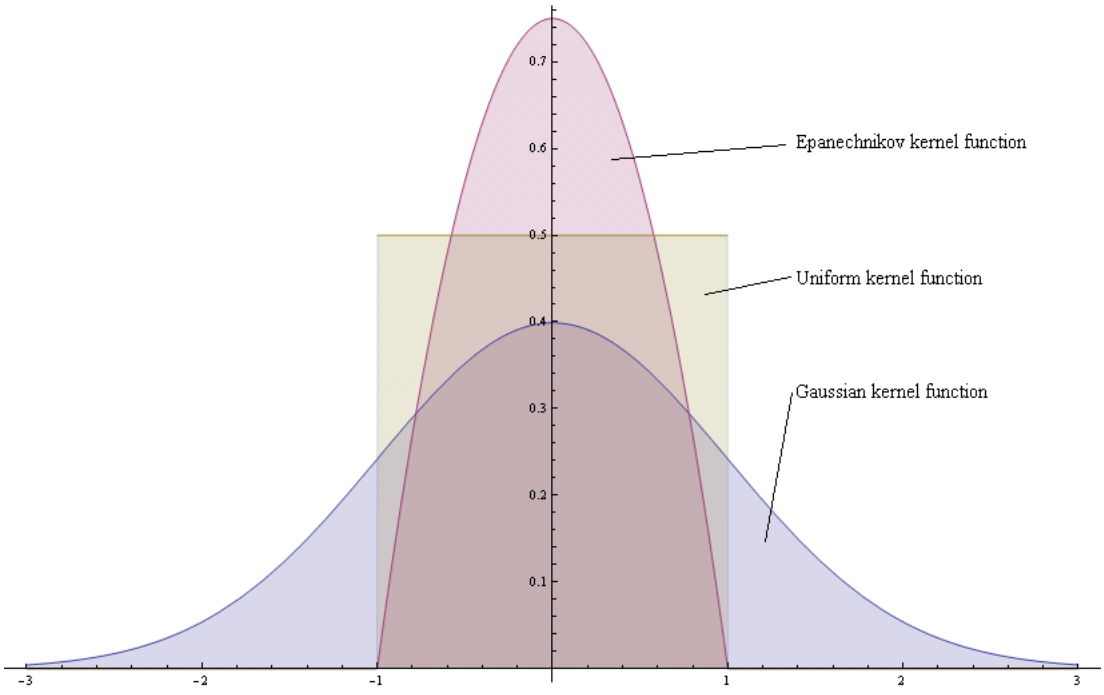


Figure 5: Gaussian, Epanechnikov and Uniform kernel function

$$Y_i = \frac{\sum_{i=1}^{N} \prod_{j=1}^{m} (h_j^c)^{-1} \cdot \kappa(\frac{X_{i,j}^c - X_{i,j}^c}{h_j^c}) \cdot Y_i}{\sum_{i=1}^{N} \prod_{j=1}^{m} (h_j^c)^{-1} \cdot \kappa(\frac{X_{i,j}^c - X_{i,j}^c}{h_j^c})} + \epsilon_i \tag{8}$$

Substituting Equation (6) in (1) yields Equation (8), a smoothing function with m unknown bandwidth parameters $(h_1^c, h_2^c, ..., h_m^c)$, the values of which determine the errors ϵ_i. Equation (5) shows that a smaller h_j^c results in a larger weight for that specific dimension of the predictor variables. If h_j^c goes to infinity, the weight goes to 0. In this way the bandwidth $\vec{h^c}$ scales the weights of each dimension of the predictor variables, but also functions as a measure for the relative importance [26] of each dimension of the predictor variables. In order to obtain a regression model with minimal error ϵ_i, it is necessary to find the optimal bandwidth. Finding the optimal bandwidth parameters (or bandwidth selection) such that the errors ϵ_i are minimal, is discussed further in Chapter 3.6.

As an example we show how the distance between cases 1, 2 and 4 from Table 1 are computed. This example, shown in Table 2, is used to illustrate the concepts described,

therefore we limit ourselves to the comparison of the complete cases based on the durations. Assume Cases 2 and 4 from Table 1 are in our data-set and Case 1 is a new case. Furthermore, assume the bandwidth h_j^c is 7 for "Test Repair" and 2 for all other variables.

Continuous weights example					
	Analyze Defect	*Repair (complex)*	*Test Repair*	*Repair (simple)*	δ^c
bandwidth	$h_1^c = 2$	$h_2^c = 2$	$h_3^c = 7$	$h_4^c = 2$	
case 1	$X_{1,1}^c = 10$	$x_{1,2}^c = 14$	$X_{1,3}^c = 10$	$X_{1,4}^c = 0$	-
case 2	$X_{2,1}^c = 5$	$x_{2,2}^c = 44$	$X_{2,3}^c = 7$	$X_{2,4}^c = 0$	-
$\frac{\kappa((X_{1,j}^c - X_{2,j}^c)/h_j^c)}{h_j^c}$	0.00876	2.76535×10^{-50}	0.051991	0.19947	2.51344×10^{-54}
case 1	$X_{1,1}^c = 10$	$X_{1,2}^c = 14$	$X_{1,3}^c = 10$	$X_{1,4}^c = 0$	-
case 4	$X_{4,1}^c = 7$	$X_{4,2}^c = 0$	$X_{4,3}^c = 9$	$X_{4,4}^c = 5$	-
$\frac{\kappa((X_{1,j}^c - X_{4,j}^c)/h_j^c)}{h_j^c}$	0.06476	4.56736×10^{-12}	0.1760	0.05641	1.46236×10^{-16}

Table 2: Example of continuous weights

With these distances the density $\sum_{i=2}^{4} \delta^c(\vec{X}_i^c, \vec{X}_1^c)$ becomes 1.46236×10^{-16}. As the weights are divided by the density, the total influence of Case 2 becomes $2.51344 \times 10^{-54} / 1.46236 \times 10^{-16} = 1.71876 \times 10^{-38}$ and the influence of Case 4 $1.46236 \times 10^{-16} / 1.46236 \times 10^{-16} = 1$. The prediction would therefore be $Y_2 \cdot 1.71876 \times 10^{-38} + Y_4 \cdot 1$, i.e. the prediction is based almost completely on Case 4. In this example, the value of "Repair(Complex)" is most responsible for the low weight for Case 2. Had the bandwidth of "Repair(Complex)" been larger, then the difference in weights would have been smaller. This shows how a small bandwidth increases the influence of a single predictor variable on the result.

In this chapter we have shown how the continuous weight function computes a weight for a case based on the measurements. In the next chapter the discrete weight function is described, that computes the weights for both categorical and ordered predictor variables.

3.5 The discrete weight function

In this section we define the discrete weight function $\delta^d : \mathcal{D}^k \times \mathcal{D}^k \to \mathbb{R}$, in which k is the total number of discrete predictor variables, l the number of discrete nominal predictor variables, and $\mathcal{D}$ is the abstract domain of the discrete predictor variable, as defined in Section 3.3. The discrete predictor variables consist of two types, namely the ordered and the nominal discrete predictor variables. The only difference between the domain of the discrete nominal predictor variables $\mathcal{D}_u \subseteq \mathcal{D}$ and the domain of the ordered discrete predictor variables $\mathcal{D}_o \subseteq \mathcal{D}$, is that there is a total order $<_{\mathcal{D}}$ defined on the domain $\mathcal{D}_o$. Note that actually, each single discrete predictor variable has an individual domain and each single discrete ordered predictor variable has an individual order defined on its domain. However for readability, we define all discrete predictor variables to have the

abstract domain $\mathcal{D}$, and all discrete ordered predictor variables to have the abstract total order $<_{\mathcal{D}}$.

Similar to $\vec{X}_i^c$ and $\vec{x^c}$, we use $\vec{X}_i^d = (X_{i,1}^d, X_{i,2}^d, \ldots, X_{i,k}^d)$ and $\vec{x^d} = (x_1^d, x_2^d, \ldots, x_k^d)$ to denote the discrete predictor variables, such that $(\forall j \in \mathbb{N} : 1 \leq j \leq k = p - m : X_{i,j}^d = X_{i,j+m} \wedge x_j^d = x_{j+m})$, i.e. the last $k = p - m$ predictor variables in $\vec{X}_i$ and $\vec{x}$ are discrete. Also it holds that $(\forall j \in \mathbb{N} : 1 \leq j \leq l : X_{i,j}^d \in \mathcal{D}_u)$ and $(\forall j \in \mathbb{N} : l+1 \leq j \leq k : X_{i,j}^d \in \mathcal{D}_o)$, i.e. the first l discrete predictor variables are categorical and the remaining $k - l$ discrete predictor variables are ordered.

The discrete weight function we use, i.e. the Li and Racine kernel [25], is applicable to both categorical and ordered discrete data. This weight function is a variation of the weight function proposed in [4] for categorical data, and although not applicable to ordered data, it is easily generalised such that it is. We show how Equation (9) (the so called Aitchison and Aitken kernel [4]) is adapted, as this provides an insight into its working. Note that in the Aitchison and Aitken kernel Equation (9) and (10) are combined into a single equation, however for readability we this format.

$$\delta^d(\vec{X}_i^d, \vec{x^d}) = \prod_{j=1}^{l} \beta^d(X_{i,j}^d, x_j^d) \tag{9}$$

$$\beta^d(X_{i,j}^d, x_j^d) = \begin{cases} 1 & \text{if } X_{i,j}^d = x_j^d; \\ h_j^d & \text{if } X_{i,j}^d \neq x_j^d \end{cases} \tag{10}$$

As with the continuous weight function, the goal of the discrete weight function is to assign a weight to a point based on the distance between $\vec{X}_i^d$ and $\vec{x^d}$. For the categorical case the distance between two unequal points is always equal, i.e. the categories either match or they don't. Also, just like in the continuous weight function there is a bandwidth $\vec{h^d} = (h_1^d, h_2^d, \ldots, h_k^d)$, such that $\big(\forall j \in \mathbb{N} : 1 \leq j \leq k : h_j^d = h_{j+m}\big)$, i.e. $\vec{h^d}$ is a k dimensional vector with the discrete bandwidth variables, as a result $\vec{h}$ is a p dimensional vector consisting of $\vec{h^c}$ and $\vec{h^d}$. The discrete bandwidth variable is limited to the domain $0 \leq h_j^d \leq 1$. Note that h_j^d is a measure of the relative importance of the current predictor variable, where smaller means more important. If the bandwidth h_j^d equals 1, the weight of the current predictor variable becomes constant for all measurements, resulting in the normal average over all measurements. In this case the predictor variable is irrelevant. If h_j^d is 0, the weight returned is 1 if $X_{i,j}^d$ is equal to x_j^d and 0 if they are unequal, which means the current predictor variable becomes an indicator.

$$\Delta_j(X^d_{i,j}, x^d_j) = \begin{cases} \left(\#a \in \mathcal{D}_o : X^d_{i,j} <_{\mathcal{D}} a <_{\mathcal{D}} x^d_j\right) & \text{if } l < j \leq k \wedge X^d_{i,j} <_{\mathcal{D}} x^d_j \wedge X^d_{i,j} \neq x^d_j \\ \left(\#a \in \mathcal{D}_o : x^d_j <_{\mathcal{D}} a <_{\mathcal{D}} X^d_{i,j}\right) & \text{if } l < j \leq k \wedge x^d_j <_{\mathcal{D}} X^d_{i,j} \wedge X^d_{i,j} \neq x^d_j \\ 1 & \text{if } 1 \leq j \leq l \wedge X^d_{i,j} \neq x^d_j \\ 0 & \text{if } X^d_{i,j} = x^d_j \end{cases} \tag{11}$$

In the ordered case the distance between two unequal variables is not always equal. We therefore define $\Delta_j : (\mathcal{D}, \mathcal{D}) \rightarrow \mathbb{N}$, a function that computes the distance between two (un)ordered discrete variables. An example of a function that computes the distance between $X^d_{i,j}$ and x^d_j is given in Equation (11), which shows that the distance between categorical variables is either 1 or 0. In Equation (11), the top two lines show that the distance between ordered variables is dependent on $<_{\mathcal{D}}$, the distance between two discrete ordered variables is equal to the distance between two natural numbers, as natural numbers are also discrete and ordered.

$$\beta^d(X^d_{i,j}, x^d_j) = \begin{cases} 1 & \text{if } X^d_{i,j} = x^d_j; \\ (h^d_j)^{\Delta_j(X^d_{i,j}, x^d_j)} & \text{if } X^d_{i,j} \neq x^d_j \end{cases} \tag{12}$$

Equation (12) shows a function that returns a weight for ordered typed variables. However, because the distance between two unequal categorical typed variables is always 1, Equation (12) returns $(h^d_j)^1$ for categorical variables, which is equal to Equation (10). Therefore, Equation (12) is suited for both ordered as well as categorical variables. Equally when the distance in both categorical and ordered typed variables is 0 (i.e. the variables are equal), one can also write $(h^d)^0 = 1$. As a result $\beta^d(X^d_{i,j}, x^d_j) = (h^d_j)^{\Delta_j(X^d_{i,j}, x^d_j)}$, which we combine into the product weight function in Equation (13). Leaving us the final version of the discrete weight function $\delta^d : (\mathcal{D}^k, \mathcal{D}^k) \rightarrow \mathbb{R}$, also known as the Li and Racine kernel [25].

$$\delta^d(\vec{X^d_i}, \vec{x^d}) = \prod_{j=1}^{k} (h^d_j)^{\Delta_j(X^d_{i,j}, x^d_j)} \quad , (\forall_j : 1 \leq j \leq k : 0 \leq h^d_j \leq 1) \tag{13}$$

Our running example contains a total of 12 different discrete variables, each of the 8 activities has an occurrence and each case has 4 data-attributes, the discrete predictor variable therefore has 12 dimensions. The measurement of each case is now a pair of a vector with predictor variables and the corresponding response variable. We want to compare the predictor variables of the partial case to the predictor variables in the measurements, but not all of the 12 discrete dimensions of the predictor variables are equally relevant. Therefore we use bandwidth $\vec{h^d}$ to adapt the influence of each individual dimension on the weight of the whole measurement in the smoothing function. As with the continuous weight function, we show a small example in Table 3. We take two occurrences and two

attributes from Table 1, specifically "Repair(complex)", "Test Repair", "phoneType", and "defectType". The bandwidths for "phoneType" is 1, for "Repair(complex)" it is 0.1 and 0.5 for the rest.

Discrete weights example					
	Repair (complex)	*Test Repair*	*phoneType*	*defectType*	δ^d
bandwidth	$h_7^d = 0.1$	$h_8^d = 0.5$	$h_9^d = 1$	$h_{10}^d = 0.5$	
case 1	$X_{1,7}^d = 1$	$X_{1,8}^d = 1$	$X_{1,9}^d = T2$	$X_{1,10}^d = 9$	-
case 2	$X_{2,7}^d = 1$	$X_{2,8}^d = 1$	$X_{2,9}^d = T2$	$X_{2,10}^d = 8$	-
$(h_j^d)^{\Delta_j(X_{1,j}^d, X_{2,j}^d)}$	1	1	1	0.5	0.5
case 1	$X_{1,7}^d = 1$	$X_{1,8}^d = 1$	$X_{1,9}^d = T2$	$X_{1,10}^d = 9$	-
case 4	$X_{4,7}^d = 0$	$X_{4,8}^d = 1$	$X_{4,9}^d = T1$	$X_{4,10}^d = 1$	-
$(h_j^d)^{\Delta_j(X_{1,j}^d, X_{4,j}^d)}$	0.1	1	0.9	0.5	0.045

Table 3: Example of discrete weights

In the beginning of this chapter we explained why it is necessary to assume that all data-attributes are categorical, as a result the data-attribute "defectType" is also treated as a categorical variable. With these weights the prediction for case 1 based on cases 2 and 4 becomes $(Y_2 \cdot 0.91743 + Y_4 \cdot 0.08257)/2$, i.e. the prediction is largely based on case 2 with a small (but noticeable) influence of case 4. As the bandwidth of "Repair(Complex)" is very low, the weight for cases with an unequal value for "Repair(Complex)" is lower, and therefore the influence of "Repair(Complex)" on the prediction is higher. The bandwidth of "phoneType", on the other hand, is very high. Therefore the influence of "phoneType" on the weight is very low. This example shows how the bandwidth has an effect on the influence of different predictor variables on the prediction. Now that all weight functions are described, the process of obtaining the optimal bandwidth is described.

3.6 Bandwidth variable optimization

Both the continuous and the discrete weight functions multiply the weights of the individual dimensions of a predictor variable in order to return a single weight, in the same way we multiply both weight functions to obtain a single weight for a mixed predictor variable. The bandwidth variables $\vec{h} = (h_1, h_2, \ldots, h_p)$ should ensure that the individual weights are scaled such that the single combined weight yields a minimum error in the regression function. By combining both the discrete and the continuous weight functions in a product, we get the final smoothing function shown in Equation (14).

$$\gamma(\vec{x}) = \frac{\sum_{i=1}^{N} \delta^c(\vec{X_i^c}, \vec{x^c}) \cdot \delta^d(\vec{X_i^d}, \vec{x^d}) \cdot Y_i}{\sum_{i=1}^{N} \delta^c(\vec{X_i^c}, \vec{x^c}) \cdot \delta^d(\vec{X_i^d}, \vec{x^d})} \tag{14}$$

The smoothing function in Equation (14) estimates the response value based on a weighted (local) average over the most similar cases in X compared to x. The bandwidth $\vec{h} \in \mathbb{R}^p$, which is defined such that $(\forall j \in \mathbb{N} : 1 \leq j \leq m : h_j = h_j^c \wedge h_j^c > 0)$ and $(\forall j \in \mathbb{N} : 1 \leq j \leq k : h_j^d = h_{j+m} \wedge 0 \leq h_j^d \leq 1)$, acts as a scale factor to determine what distance is accepted as "close" and thus should receive a higher weight. Furthermore if h_j receives a value that is relatively extreme it also serves as a measure of the importance of the j^{th} (discrete or continuous) predictor variable, as shown in [26].

In both the discrete and the continuous case a bandwidth close to 0 indicates a predictor variable that is important. If the bandwidth for a predictor variable is very small, cases that are "distant"' receive a very low weight, whereas cases that are "close" receive a much higher weight. Also, for the continuous weight function the distance that is called "close" is smaller as the bandwidth is smaller. Therefore a small bandwidth means a predictor variable has a very big influence on which cases receive a high or low weight, as cases that are not "close" receive a very low weight.

Besides the minimum of 0, there is also a maximum value of 1 for the discrete bandwidth of the discrete weight function. If the discrete bandwidth equals 1 it results in a constant weight of 1 independent of the value of $X_{i,j}^d$, as the weights are multiplied, the weight of 1 is irrelevant. Therefore, a bandwidth equal to 1 indicates a discrete predictor variable that does not influence the prediction.

However, the bandwidth for the continuous weight function has only the theoretical maximum of ∞. As the bandwidth for a specific continuous predictor variable increases, the distance between values that is considered "close" increases. If the bandwidth is large enough that all cases are considered "close" for this predictor variable, all cases receive a high weight. If the difference in weights for cases is small their influence is also small. Thus, only if a continuous bandwidth is larger than the variation in cases, the predictor variable is irrelevant. It is therefore not straightforward to determine if a continuous predictor variable is irrelevant. More on this topic is found in [26].

As the bandwidth $\vec{h}$ can increase or decrease the effect of certain components of $\vec{X}_i$ there is an asymptotically optimal [33] $\vec{h}'$ such that $\epsilon_i = Y_i - \gamma(\vec{X}_i)$ is minimal, assuming there is some relationship between the predictor variables $\vec{X}_i$ and the response variable Y_i. By taking the *mean square error* as our error-measurement, we derive the target function in Equation (15) needed to find an optimal $\vec{h}'$. By minimising MSE_γ over $\vec{h}$ we find the optimal $\vec{h}'$ such that the mean square error is minimal, meaning we have a γ that models the relationship between Y_i and $\vec{X}_i$ as accurately as possible.

$$MSE_\gamma = (N)^{-1} \sum_{i=1}^{N} (Y_i - \gamma(\vec{X}_i))^2 \tag{15}$$

There is one problem with this; by using MSE_γ we estimate a Y_j by $\gamma(\vec{X}_j)$ which in turn

compares this $\vec{X}_j$ to all measurements in $(Z_i, \vec{X}_i, Y_i)$, thus comparing $\vec{X}_j$ to $\vec{X}_j$. To limit this type of bias we use a technique called *leave-one-out cross validation*, which means we take $\{(Z'_i, \vec{X}'_i, Y'_i)\}_{i=1}^{N-1} = \{(Z_i, \vec{X}_i, Y_i)\}_{i=1}^{N} \setminus (Z_j, \vec{X}_j, Y_j)$. In this way the estimation for Y_j is computed by a $\gamma'(\vec{X}_j)$ that compares $\vec{X}_j$ to each measurement in $\{(Z'_i, \vec{X}'_i, Y'_i)\}_{i=1}^{N-1}$.

$$\gamma'(X_i) = \frac{\sum_{j=1, j\neq i}^{N} \delta^c(\vec{X}_j^c, \vec{X}_i^c) \cdot \delta^d(\vec{X}_j^d, \vec{X}_i^d) \cdot Y_j}{\sum_{j=1, j\neq i}^{N} \delta^c(\vec{X}_j^c, \vec{X}_i^c) \cdot \delta^d(\vec{X}_j^d, \vec{X}_i^d)} \tag{16}$$

Minimizing the $MSE_{\gamma'}$ over $\vec{h}$ provides us with the optimum $\vec{h}$ with an minimal mean square error. Note however, that this minimization becomes very complex, meaning it takes a long time to compute. In a realistic problem there might be V variables (both discrete and continuous), with for example N cases. It is easy to see that the complexity of a single calculation of $MSE_{\gamma'}$ is $\Theta(MSE_{\gamma'}) = N^2V$. In order to find the complexity of the bandwidth selection, the minimization must vary over the domain of $\vec{h} = (h_1, h_2, \ldots, h_p)$, which is $(\forall j \in \mathbb{N} : 1 \leq j \leq k : 0 < h_j \leq \infty)$ and $(\forall j \in \mathbb{N} : m+1 \leq j \leq p : 0 \leq h_j \leq 1)$. Because we can not iterate over an infinite number of values, we must limit this to a discrete set of values. Assume $\mathfrak{D}$ is the domain size of $\vec{h}$, the more precise we want the minization to be, the more values we must put in $\mathfrak{D}$. Given $\mathfrak{D}$, first the minimization must vary each component of $\vec{h}$ over its domain and then it must vary each component V times in order to find all possible combinations, thus the total complexity of finding the bandwidth becomes $\theta(N^2V^2\mathfrak{D}^V)$. This means the bandwidth minimization is of exponential complexity.

To ensure the bandwidth selection is computed within a feasible time, we use a subset of the measurement during the bandwidth selection. This subset is of size $\frac{N}{S}$, i.e. the subset is S times smaller than N. Given a set of measurements $(Z_i, \vec{X}_i, Y_i)$ for $0 \leq i \leq N$, we define a subset of those measurements $(Z'_s, \vec{X}'_s, Y'_s)$ for $0 \leq s \leq (\frac{N}{S})$, which we use with Equation (16) to find an optimum bandwidth. This way the complexity of the bandwidth selection becomes $\theta((\frac{N}{S})NV^2\mathfrak{D}^V)$, which is a factor $\frac{1}{S}$ smaller. In order to ensure the bandwidth found using the subset of measurements is still a good bandwidth for the complete set of measurements, we must try to balance the subset of measurements such that they still present a good representation of the process. In the next chapter, we show that in order to do this, a single case is split up into measurements representing that case over time. If we randomly select cases and put all measurements of a case into the subset of cases for the bandwidth selection, we ensure that we have a balanced set of measurements for the bandwidth selection.

Much work on optimizing the bandwidth selection exists. For this thesis we have chosen not to research this topic in-depth, as a software package we use provides optimized methods for this [19, 20]. We therefore refer to [18, 21] or [31] for more information on this topic.

In summary, we now have a regression model that is capable of using both continuous, discrete nominal and discrete ordinal predictor variables. The regression model uses a

bandwidth $\vec{h}$, which we need to optimize once for a set of predictor variables using γ', such that the regression model has a minimal MSE over the cross-validated set of historical measurements $\{(Z_i, \vec{X}_i, Y_i)\}$. After the optimal bandwidth is found, we have a regression model γ that can be applied to a new partial case. In the next chapter we describe how measurements, that lie at the basis of the prediction method, are obtained.

4 Non-parametric regression as a predictor

In Section 1.3 we explained that the prediction method presented in this thesis only uses the data that is in the event-log. In the previous chapter we explain that non-parametric regression constructs a local average for the prediction of a new partial case such that it is based on the most similar cases. In this chapter we show how we derive the measurements which we use with non-parametric regression in order to make predictions for (partial) cases in information systems. Section 4.1 continues with the running example, which was introduced in Section 3.1, and Section 4.2 presents some preliminaries to ensure all readers are familiar with the syntax used in this thesis. The preliminaries can be skipped by readers familiar with these notations.

4.1 The running example

The running example introduced in Section 3.1 was used to illustrate our approach. So far we kept the running example simple to illustrate how non-parametric regression works. In this chapter we show the running example in more detail, starting with the log. The log used is stored in MXML format, which is a XML standard for storing event-logs. A fragment of a log is shown in Figure 6, this example shows the "start"- and "complete"-events related to the activity "Test Repair", and the complete-event relating to "Analyze Defect". The events have a time-stamp, using the time-stamps of the "start"- and "complete"-events we compute the duration of an activity. Table 4 lists all events of the case shown in Figure 6.

Timestamp	**Event Type**	**Element**	**Attributes**
01.01.1970 09:38:00.000	complete	Register	-
01.01.1970 09:38:00.000	start	Analyze Defect	-
01.01.1970 09:48:00.000	complete	Analyze Defect	defectType: 9 phoneType: T2
01.01.1970 10:04:00.000	start	Repair(Complex)	-
01.01.1970 10:18:00.000	complete	Repair(Complex)	-
01.01.1970 10:18:00.000	start	Test Repair	-
01.01.1970 10:28:00.000	complete	Test Repair	defectFixed: true numberRepairs: 0
01.01.1970 10:32:00.000	complete	Inform User	-
01.01.1970 10:35:00.000	complete	Archive Repair	defectFixed: true numberRepairs: 0

Table 4: Events of the case corresponding to the fragment of eventlog in Figure 6

Table 4 shows the sequence of events for a single case, and as the case progresses more information becomes available. Also, as the case progresses the total remaining cycle time decreases. As a result, the predictor variables and the corresponding response variable

```
...
<Process id="DEFAULT" description="Simulated process" >
...
<ProcessInstance id=...>
...
   <AuditTrailEntry>
      <Data>
         <Attribute name="defectType">9</Attribute>
         <Attribute name="phoneType">T2</Attribute>
      </Data>
      <WorkflowModelElement>Analyze Defect</WorkflowModelElement>
      <EventType>complete</EventType>
      <Timestamp>1970-01-01T09:48:00.000+01:00</Timestamp>
      <Originator >Tester4</Originator>
   </AuditTrailEntry>
...
   <AuditTrailEntry>
     <WorkflowModelElement>Test Repair</WorkflowModelElement>
     <EventType>start</EventType>
     <Timestamp>1970-01-01T10:18:00.000+01:00</Timestamp>
     <Originator>Tester5</Originator>
   </AuditTrailEntry> <AuditTrailEntry>
     <Data>
       <Attribute name="numberRepairs">0</Attribute>
       <Attribute name="defectFixed">true</Attribute>
     </Data>
     <WorkflowModelElement>Test Repair</WorkflowModelElement>
     <EventType>complete</EventType>
     <Timestamp>1970-01-01T10:28:00.000+01:00</Timestamp>
     <Originator>Tester5</Originator>
   </AuditTrailEntry>
...
</ProcessInstance>
...
</Process>
```

Figure 6: Fragment of an MXML log corresponding to case 1

for a case change at each event. If we use only the predictor and response variables of a complete case, we lose a great deal of information. Therefore, the measurements of a case are based on the *prefixes* of that case, meaning that for a single case, we derive a measurement after each "complete"-event in that case. This is best explained by Table 5, which shows the measurements that correspond to the events of the single case shown in Table 4. Note that if the average case consists of 8 "complete"-events, and there are 1000 cases in the log, there are about 8000 measurements.

In Table 5 every column of values is a single measurement, each measurement corresponds to a prefix of the case shown in Table 4. In other words; each measurement $\mathcal{M}_i$ in Table

Measurements of case 1		$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$	$\mathcal{M}_4$	$\mathcal{M}_5$	$\mathcal{M}_6$
response variable	remaining cycle time	57	47	17	7	3	0
continuous prediction variable	Analyze Defect Duration	0	10	10	10	10	10
	Repair (complex) Duration	0	0	14	14	14	14
	Test Repair Duration	0	0	0	10	10	10
	Repair (simple) Duration	0	0	0	0	0	0
categorical prediction variable	defectType attribute	-	9	9	9	9	9
	defectFixed attribute	-	-	-	true	true	true
	phoneType attribute	-	T2	T2	T2	T2	T2
	numberRepairs attribute	-	-	-	0	0	0
ordered prediction variable	Register Occurrence	1	1	1	1	1	1
	Analyze Defect Occurrence	0	1	1	1	1	1
	Repair (complex) Occurrence	0	0	1	1	1	1
	Test Repair Occurrence	0	0	0	1	1	1
	Repair (simple) Occurrence	0	0	0	0	0	0
	Inform User Occurrence	0	0	0	0	1	1
	Archive Repair Occurrence	0	0	0	0	0	1

Table 5: Example of measurements of the case in Table 4

5 contains the response variable and predictor variables as they are after each "complete" event in Table 4. The measurements in Table 5 show how a single case generates multiple measurements. In this table we have taken the total remaining cycle time as the response variable, however, it is easy to use other response variables as we show later. Assume that we have a new partial case for which we want to make a prediction. First, we derive the single measurement m of that partial case. The prediction for this single measurement m is computed by the non-parametric regression, which constructs a local average such that the prediction is based on the weighted average of the response variable of the most similar (old) measurements $\mathcal{M}$. In other words, we use the measurements in $\mathcal{M}$ that are most similar to m to construct a prediction for m. If we compare the measurement m of a partial case to the measurements in $\mathcal{M}$, we find that the most similar measurements are those that correspond to a prefix that has the same length as the partial case has. In other words, we compare a new partial case to the partial/prefixed cases that correspond to the full cases from our log.

4.2 Preliminaries

Let $n \in \mathbb{N}$. A *sequence* over S of length n is a function $\sigma : \{1, .., n\} \to S$. If $n > 0$ and $\sigma(0) = a_1, .., \sigma(n) = a_n$, we write $\sigma = \langle a_1, .., a_n \rangle$, and σ_i for $\sigma(i)$. The length of a sequence is denoted by $|\sigma|$. The sequence of length 0 is called the empty sequence, and is denoted by $\langle\rangle$. The set of finite sequences over S is denoted by S^*. Let $\upsilon, \tau \in S^*$ be two sequences. Concatenation, denoted by $\sigma = \upsilon \cdot \tau$ is defined as $\sigma : \{1, ..., |\upsilon| + |\tau|\} \to S$, such that for $1 \leq i \leq |\upsilon|$, $\sigma(i) = \upsilon(i)$, and for $|\upsilon| < i \leq |\sigma|, \sigma(i) = \tau(i - |\upsilon|)$.

Furthermore, the prefix from index i to j on sequences is defined by $v' = \downarrow_{i,j}(v)$, such that if $i \geq j$, then $v' = \langle\rangle$, otherwise $v' = \langle v_i, \ldots, v_{j-1}\rangle$, i.e. for all sequences v holds that $\downarrow_{0,|v|}(v) = v$. The term $\vec{x}$ is used to denote column vectors and for a sequence $\sigma \in S^*$, the *Parikh vector* $\vec{\mathcal{P}}(\sigma)\colon S \to \mathbb{N}$ defines the number of occurrences of each element of S in the sequence, i.e. $\vec{\sigma}(s) = |\{i \mid 1 \leq i \leq |\sigma| \wedge \sigma(i) = s\}|$, for all $s \in S$.

Let X be a set and $<_X$ a total order on this set, we define $pref : \mathcal{P}(X) \to \mathcal{P}(\mathcal{P}(X))$ to be a function such that for $s \in \mathcal{P}(X)$ holds that $pref(s) = \{\{e' \in s \mid e' <_X e \vee e' = e\} \mid e \in s\} \cup \{\emptyset\}$, i.e. a function that returns all prefixes for a given set X of totally ordered elements.

Let $f : A \to B$ be a function. For $X \subseteq A$, we define $f(X) = \{f(x) \mid x \in X\}$, and for $b \in B$ we define $f^{-1}(b) = \{a \in A \mid f(a) = b\}$.

4.3 Cases, logs, measurements

In order to explain the measurements in detail, we start by formalising certain concepts such as the definition of a log. Using the definition of a log we describe various other functions that are used to derive measurements, as well as the measurements themselves.

Definition 4.1 (Log)**.**
Let $W = (C, A, AI, E, AN, AV, time, ai, act, case, case\_attval, event\_attval)$ be a log where:

- C	is the set of case identifiers
- A	is the set of activity identifiers
- AI	is the set of activity instances
- E	is the set of events
- AN	is the set of attribute names
- AV	is the set of attribute values
- $time : E \to \mathbb{R}_0^+$	gives the time of an event
- $ai : E \to AI$	gives the activity instance where an event refers to
- $act : AI \to A$	maps activity instances onto activities
- $case : AI \to C$	maps activity instances onto cases
- $case\_attval : (C \times AN) \to (AV \cup \{\perp\})$	maps a case and attribute name onto an attribute value
- $event\_attval : (E \times AN) \to (AV \cup \{\perp\})$	maps an event and an attribute name onto an attribute value

such that
- E is totally ordered by relation $<_E$
- $(\forall_{e_1,e_2 \in E} : e_1 <_E e_2 \Rightarrow time(e_1) \leq time(e_2))$
- $ai, time, act, case, case\_attval$ and $event\_attval$ are total functions
- "*eventtype*" $\in AN \wedge \{$"*start*", "*complete*"$\} \subseteq AV$
- $(\forall e \in E : event\_attval(e,$ "*eventtype*"$) \in \{$"*start*", "*complete*"$\})$
- $\perp$ denotes an empty attribute value, which is needed to make the functions total

In other words, a log contains a list of cases. The cases consist of a set of activity instances, each activity instance corresponds to one or more events, and each activity instance also corresponds to an activity. The events and cases contain attributes, such that all events contain an attribute "eventtype" that has a value "start" or "complete". Furthermore, all events are ordered in time.

Definition 4.2 (Measurement Related Functions)**.** Let $c \in C$, $es \subseteq E$, $a \in A$, and $an \in AN$, such that $(\forall e \in es : case(ai(e)) = c)$. We define :

$$
\begin{array}{rcl}
is\_complete & : & E \rightarrow \mathbb{B} \\
is\_complete(e) & = & (event\_attval(e, \text{"}eventtype\text{"}) = \text{"}complete\text{"}), \text{ for } e \in E \\
start & : & \mathcal{P}(E) \rightarrow \mathbb{R}_0^+ \\
start(es) & = & \min\limits_{e \in es} \quad time(e) \\
end & : & \mathcal{P}(E) \rightarrow \mathbb{R}_0^+ \\
end(es) & = & \max\limits_{e \in es} \quad time(e) \\
duration & : & \mathcal{P}(E) \rightarrow \mathbb{R}_0^+ \\
duration(es) & = & end(es) - start(es) \\
count & : & A \times \mathcal{P}(E) \rightarrow \mathbb{N} \\
count(a, es) & = & |\{e \in es | is\_complete(e) \wedge act(ai(e)) = a\}| \\
last\_att & : & \mathcal{P}(E) \times AN \nrightarrow E \\
last\_att(es, an) & = & e, \text{such that } e \in es \wedge (\bot \neq event\_attval(e, an)) \wedge \\
 & & \qquad \neg (\exists e' \in es : e' >_E e : (\bot \neq event\_attval(e', an)) \\
att\_val & : & \mathcal{P}(E) \times AN \rightarrow AV \\
att\_val(es, an) & = & \begin{cases} case\_attval(c, an), \text{iff } \neg (\exists e \in es : \bot \neq event\_attval(e, an)) \\ event\_attval(last\_att(es, an), an), \text{otherwise} \end{cases}
\end{array}
$$

The function $is\_complete(e)$ returns true if the event is of type "complete", $start(es)$ and $end(es)$ return the time of the first and the last event for a set of events. The function $duration(es)$ returns the difference between the time of the first and the time of the last event in a set of events, or in other words the duration. The function $count(a, es)$ returns the number of events of type complete in the set es that correspond to activity a, i.e. the number of times activity a was completed in a set of events. This definition of *count* means that we only count an activity if it was completed. The function $att\_val(es, an)$ returns the last recorded value of the attribute an, or $\bot$. The last recorded value is the value of the last event that contained a value for attribute an, or the value of the case attribute if no event contains a value for attribute an. In att_val a function $last_att(es, an)$ is used, that is only used here, this function returns the last event that contains a value for attribute an, under the assumption that this event exists. Using these functions we construct the remaining functions that derive the measurements from a log.

In Chapter 3 we showed that the non-parametric regression requires a set of measurements $\{(Z_i, \vec{X}_i, Y_i)\}_{i=1}^{N}$, where Z_i is a unique identifier for each tuple, $\vec{X}_i \in \mathcal{C}^p$ is a vector with p predictor variables and $Y_i \in \mathbb{R}$ is the corresponding response variable. In other words, the non-parametric regression requires a set with tuples, that contain a unique identifier, a vector with predictor variables and a corresponding response variable. So far we left Z_i undefined, however, in Definition 4.3 we define a prefix es' of a set of events es that belong to a single case as the unique identifier. Because each es belongs to a single case each es is unique, and as a result each prefix es' is also unique.

Definition 4.3 (Measurements)**.** Let $W = (C, A, AI, E, AN, AV, time, ai, act, case, case\_attval, event\_attval)$ be a log, and $\mathcal{C}^p$ the domain of predictor variables. The set of measurements $\mathcal{M}$ given a function $pred\_vars : \mathcal{P}(E) \rightarrow \mathcal{C}^p$, that returns a vector with predictor variables, and a function $resp\_var : \mathcal{P}(E) \times \mathcal{P}(E) \rightarrow \mathbb{R}_0^+$, that returns a response variable, is :

$$\bigcup_{c \in C} \left\{ (es', pred\_vars(es'), resp\_var(es, es')) | es = ai^{-1}(case^{-1}(c)) \wedge es' \in pref(es) \right\} \quad (17)$$

In Definition 4.3, es' denotes a prefix of the set of events (or case) es. As a result $pred\_vars(es')$ are the predictor variables corresponding to that prefix, and $resp\_var(es, es')$ is the response variable that corresponds to that prefix es' of the case es. Note that the function *pref* was defined in the preliminaries, and returns all prefixes of a set of events. In Definition 4.3 we can exchange $resp\_vars$ for a different function so that we are able to use different response variables, in Section 4.5 we define three versions of $resp\_vars$. We need to define only a single version $pred\_vars$, because this function is able to return a vector with any combination of the three types of predictor variables.

4.4 Predictor variables

In Chapter 3 we distinguished three different types of predictor variables, namely the number of occurrences of activities, the duration of activities, and the data-attributes. The prediction method allows for any combination of these predictor variables in the measurements such that Equation (4) holds, in other words, the predictor variables are required to be a vector such that the first m variables are continuous, the following l variables are categorical and the last $k - l$ variables are ordered. If m and l are 0 this results in a vector with only activity occurrences, similarly if l is 0 we obtain a vector with both activity durations and activity occurrences as predictor variables, in total we have eight possible combinations of predictor variables. In the following section we define the predictor variables for the activity durations, the activity occurrences and the attributes. These three types of predictor variables are be combined into a single vector of size $p = m + k$ such that Equation (4) holds.

Definition 4.4 (Activity Duration Predictor Variables)**.** For $es' \in \mathcal{P}(E)$, and $m \in \mathbb{N}$, let $asd : \{1, \ldots, m\} \rightarrow A$ be a bijection that maps the numbers 1 to m to activities and vice versa, i.e. an index of m activities. We define the function $dur\_pred\_vars(es') : \mathcal{P}(E) \rightarrow \mathbb{R}^m$ such that it returns a vector with the activity durations of size m. We define $dur\_pred\_vars(es') = (pv_1, pv_2, \ldots, pv_m)$, for which we define $pv_i \in \mathbb{R}$ as:

$$pv_i = \frac{\sum_{as \in act^{-1}(asd(i))} duration\left(ai^{-1}\left(as\right) \cap es'\right)}{count(asd(i), es')} \tag{18}$$

The function that returns the activity durations in Definition 4.4 computes the average duration of an activity. The duration of each activity instance is computed as the difference between the first and the last event corresponding to that activity instance of the activity $asd(i)$ in es', which means that if an activity instance has both a "start"- and a "complete"-event it has a duration and if it only has a "complete" event it has a duration of 0. In Definition 4.4 the bijection $asd : \{1, \ldots, m\} \rightarrow A$ acts both as a selector for which activity durations are used as predictor variables, as well as an index to map the activity names to numbers. If no $i \in \mathbb{N}$ exists such that $asd(i) = a$ then the duration of activity a is not in the predictor variables.

Definition 4.5 (Attribute Predictor Variables)**.** For $es' \in \mathcal{P}(E)$, and $l \in \mathbb{N}$, let $ans : \{m+1, \ldots, m+l\} \rightarrow AN$ be a bijection that maps l natural numbers to attribute names, i.e. an index of attribute names. We define the function $att\_pred\_vars(es') : \mathcal{P}(E) \rightarrow (\mathcal{D}^u)^l$ such that it returns a vector with data-attributes of size l, as $att\_pred\_vars(es') = (pv_1, pv_2, \ldots, pv_l)$, where $pv_i = att\_val(es', ans(i+m))$.

In Definition 4.5 we use a bijection ans similar to asd. Next we define the predictor variables for activity occurrences in Definition 4.6, in which we use aso a bijection that is also similar to asd as a selector for the activity occurrences that are used in the predictor variables. Note that the occurrences are discrete ordered predictor variables, meaning that $\mathbb{N} \subseteq \mathcal{D}^o$.

Definition 4.6 (Activity Occurrence Predictor Variables)**.** For $es' \in \mathcal{P}(E)$, and $k, l \in \mathbb{N}$, let $aso : \{m+l+1, \ldots, m+k\} \rightarrow A$ be a bijection that maps $k-l$ natural numbers to activities and vice versus, i.e. an index of activities. We define the function $occ\_pred\_vars(es') : \mathcal{P}(E) \rightarrow (\mathcal{D}^o)^{k-l}$ such that it returns a vector with the activity occurrences of size $k-l$. We define $occ\_pred\_vars(es') = (pv_1, pv_2, \ldots, pv_{k-l})$, where $pv_i = count(es', aso(i+m+l))$.

With these definitions various combinations of different types of predictor variables can be used such that Equation (4) holds. We vary what attributes and activities are used in the predictor variables by using different functions for asd, aso, and ans. Note that m is the number of activity durations mapped by asd, l the number of attributes mapped by ans and $k-l$ the number of activity occurrences mapped by aso. Using definitions 4.4,

4.5 and 4.6 we define the combination of all predictor variables in Definition 4.7, such that Equation (4) holds.

Definition 4.7 (Combined Predictor Variables)**.** For $es' \in \mathcal{P}(E)$, and $p, k, l, m \in \mathbb{N}$, such that $p = m + k$ and $l \leq k$. We define the function $pred\_vars(es') : \mathcal{P}(E) \rightarrow \mathcal{C}^p$ such that $pred\_vars(es') = (pv_1, pv_2, \ldots, pv_p)$, where

$$pv_i = \begin{cases} count(asd(i), es')^{-1} \sum_{as \in act^{-1}(asd(i))} duration\left(ai^{-1}\left(as\right) \cap es'\right) & \text{,iff } 1 \leq i \leq m \\ att\_val(es', ans(i)) & \text{,iff } m < i \leq m + l \\ count(es', aso(i)) & \text{,iff } m + l < i \leq p \end{cases} \tag{19}$$

The example measurements shown in Table 5 are obtained using a *asd* such that it returns (in this order) "Analyze Defect", "Repair (complex)", "Test Repair", and "Repair (simple)". Furthermore a single measurement of Table 5 is a tuple with a unique identifier $\mathcal{M}_i$, a vector with predictor variables, and a response variable, such that for the predictor variables $\vec{x_i}$ it holds that $(\forall j \in \mathbb{N} : 1 \leq j \leq 4 : x_{i,j} \in \mathbb{R}) \wedge \forall j \in \mathbb{N} : 5 \leq j \leq 8 : x_{i,j} \in \mathcal{D}^u) \wedge \forall j \in \mathbb{N} : 9 \leq j \leq 15 : x_{i,j} \in \mathcal{D}^o)$.

In the next section we define the different response variables, with which we control what we predict.

4.5 Response variables

There are three different types of response variables, the total remaining cycle time, the remaining cycle time until activity a, and the occurrence of activity a. In this section we define three versions of the function $resp\_vars : (\mathcal{P}(E) \times \mathcal{P}(E)) \rightarrow \mathbb{R}_0^+$, one for each type of response variable. By using the correct version of $resp\_vars$ in Definition 4.3, the measurements for each type of prediction are obtained.

Definition 4.8 (Total Remaining Cycle Time)**.** $trct\_resp\_vars(es, es') = end(es) - end(es')$

Definition 4.8 is a function that returns the difference between the last event in the partial case es' and the last event in the complete case es, i.e. the total remaining cycle time. In Definition 4.9 we define the remaining cycle time until activity a, a function that returns the difference between the time of the first "complete"-event of activity a in the complete case es, and the time of the last event in the partial case es'. However, if there is no occurrence of the activity a in the measurement it is impossible to calculate a time until that activity. Also, because the non-parametric prediction method is capable of predicting the occurrence of an activity a, we assume for the prediction of the remaining cycle time until a that the activity a will in fact occur. Therefore, we filter the cases such that only cases that have an occurrence of activity a are used in the measurements. Equation (20) is the alternative equation for Definition 4.3 used to construct the measurements. Note that

the remaining cycle time until an activity a becomes negative if the activity has already occurred, therefore we ensure that $resp\_vars$ never returns a value lower than 0.

$$\bigcup_{c \in C} \{(es', pred\_vars(es'), rcta\_resp\_var_a(es, es')) | es = ai^{-1}(case^{-1}(c)) \wedge a \in act(case^{-1}(c)) \wedge es' \in pref(es)\} \quad (20)$$

Definition 4.9 (Remaining Cycle Time until a)**.** $rcta\_resp\_vars_a(es, es') = \max(start(\{e \in es \mid is\_complete(e)\} \cap ai^{-1}(act^{-1}(a))) - end(es'), 0)$

The total amount of occurrences of activity a in a sequence is the only response variable that is equal for all measurements of a single case, i.e. the total amount of occurrences of an activity in a case does not change as the case progresses. Note that the occurrence predictor uses Definition 4.3, i.e. the occurrence predictor uses all cases and not just those with an occurrence of activity a.

Definition 4.10 (Occurrence of a)**.** $occ\_resp\_vars(es, es') = count(a, es)$.

In Definition 4.10 the occurrence response variable is not a continuous variable but a natural number. At first sight this is not a problem, however, the regression functions in equations (14) and (16) return a continuous variable, which means that the regression functions can predict e.g. 0.56 occurrences. Therefore we need to round equations (14) and (16), such that they return a natural number. Because the bandwidth selection uses the rounded Equation (16), we ensure that a bandwidth is found such that the rounded Equation (14) has a minimal error.

4.6 Summary of the prediction method

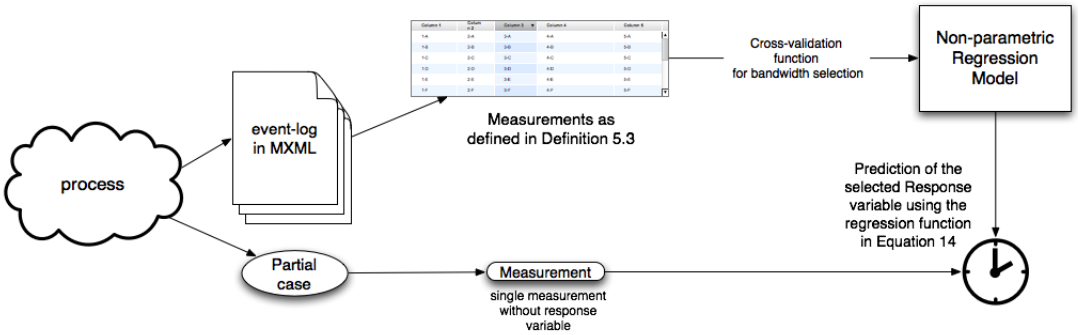


Figure 7: Detailed overview of the prediction method

Now that all math and functions necessary have been described, we summarize the prediction method, and show how they are combined. Initially there is an information system

that supports so business process. This information system collects data on the cases that are handled. An event-log in MXML is constructed based on the data in the information system, using Definition 4.3 we can construct measurements that contain predictor variables using Definition 4.7, and one of three possible response variables $resp\_vars$ (the total remaining cycle time in Definition 4.8, the total cycle time until activity a in Definition 4.9, and the occurrence of activity a in Definition 4.10). The resulting measurements $\mathcal{M} \in (\mathcal{P}(E) \times \mathcal{C}^p \times \mathbb{R}_0^+)$ can be used to construct a non-parametric regression model.

Section 3.6 described how the non-parametric regression model can be created by minimizing $MSE_{\gamma_h'}$ (Equation (15)), the mean square error of the cross-validated regression model, over the bandwidth h. This is an expensive computation in terms of time, but is required only once for a set of measurements. As a result we find an optimum bandwidth h such that the cross-validated regression model has a minimum error ϵ. Now we have a regression model $\gamma : \mathcal{C}^p \times \mathbb{R} \rightarrow \mathbb{R}$ (Equation (14)) that uses the optimum bandwidth, the measurements $\mathcal{M}$ and the single measurement of the partial case to make a prediction for that partial case. The type of prediction we make is dependent on which response variable is used in the measurements.

This prediction method is capable of handling mixed sets of predictor variables, more specifically we can use continuous, discrete ordinal (or ordered), and discrete nominal (or categorical) types of data. This prediction method can be used to predict both remaining cycle time and occurrences of activities, and it requires no assumptions on the underlying process to do so.

5 Validation

To validate our method it is necessary to verify where the prediction method behaves as expected, and to test the performance of the prediction method. With the help of simulations we verify specific aspects of processes that either increase or decrease the performance of the prediction method, such as the effect of deferred or exclusive choices on the flow in a process. To show that our prediction method performs better than the simple average estimators mentioned in the introduction, a case study has been conducted. In Section 5.1 we describe this case study, in which the non-parametric prediction method is tested on a case taken from practice. In Section 5.1 we show a set of simulations that is used to verify certain properties of the method.

5.1 Case study

To test the performance of our prediction method, we applied it to a log taken from practise. As a benchmark we used simple average estimators mentioned in the introduction and we show that out approach outperforms these estimators. We used a dataset called "bezwaar WOZ" from a Dutch municipality [23]. The process described in this log is the process of handling objections filed against real estate taxes. A model of the process is shown in Appendix C on page xv.

5.1.1 Experiment setup

Case description From the log, that originally contained 1982 cases, we only kept those cases of which both their first and their last activity were performed in the measurement period. Furthermore, we cleaned the log such that the activities not relating to the main procedure were contained less outliers. This resulted in a log containing 706 cases, which were handled by the municipality between February 28^{th} 2005 and November 8^{th} 2005 (a period of 252 days). In total, 9218 events were recorded, relating to the start and completion of 12 activities. Note that the start-events were only used to obtain the durations of each activity, i.e., only "complete" events were used in the measurements. Furthermore, all cases and all events were annotated with data attributes, which we all used in the analysis.

Method By using a 10-fold cross-validation, we used 90% of the log as the set of cases on which we base our measurements. Note that the measurements consist of the prefixes of the cases in the log, as described in Chapter 4. The prefixes of the cases in the remaining 10% of the log are then used as "partial" cases for which we make a prediction. This way we predict the remaining cycle time for each prefix of each case in the log without bias. We have predicted the total remaining cycle time, the occurrence of the activity called "0Z12 Hertaxeren", and the remaining cycle time until the activity "0Z12 Hertaxeren".

For all three types of prediction, we have tested the average predictor, the nonparametric predictor with only occurrences, with only data-attributes, with only activity-durations, and with all three (occurrences, data-attributes and activity-durations) combined.

Hardware For each of the regression experiments, the optimal values for the bandwidth parameters were calculated using a package called NP [19], which includes an internal cross-validation for finding the optimal values of the bandwidth parameters and is an implementation of the non-parametric regression described in [25]. Bandwidth selection and the computation of the predictions for each partial prefix were done using the software package R [20], running on four dual quad-core 2.66GHz Intel Xeon CPUs with 16 GB of memory each. This setup allowed us to run the 10 experiments of the 10-fold cross validations in parallel, as each experiment uses a single thread. A detailed overview of the experiment setup is given in appendix A on page ii of the Appendix.

Benchmark A straightforward way to predict the remaining cycle time, is to make a prediction based on the average. As mentioned in the introduction, this works by taking the average total cycle time minus the time spent waiting so-far. It is common to add a certain confidence interval to this prediction, such that the predicted value becomes a range of likely values. As the average is based on all cases, an average prediction with a confidence interval of for example 95% results in a predicted range that most likely contains the actual value. As we focus on the accurate estimation of a single value rather than a range, we use the average prediction without a confidence interval.

The average predictor for the remaining cycle time works assuming each case has a total cycle time equal to the average total cycle time over all cases. The average predictor for the discrete case, the average occurrence estimator, works in the same way. As the occurrences of a certain activity a can be 0, 1, and in case of loops 2 or more, the average amount of occurrences based on all cases becomes the prediction for each case. Thus if, on average, a case contains the occurrence of activity a once, the average occurrence predictor predicts 1 for all cases.

5.1.2 Total remaining cycle time prediction

Results of the total remaining cycle time prediction

For each experiment we recorded the estimates and the actual values. These were used to calculate the absolute error, the relative error and the mean square error for the complete experiment. The overall mean square errors are shown in Table 6. A scatterplot with the average based predictions, the nonparametric predictions and the actual values is shown in Figure 8. In this scatterplot the X-value denotes the time spent at the time of the prediction, and the Y-value denotes the (predicted or actual) total remaining cycle time.

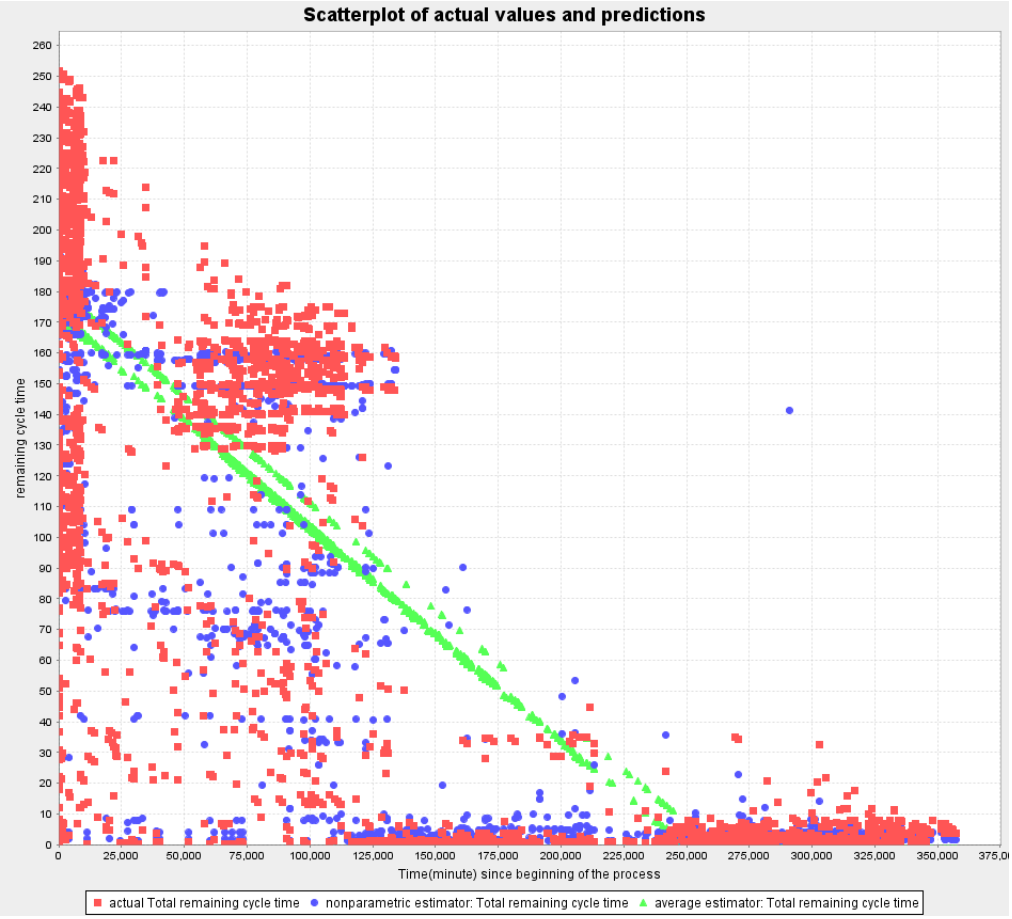


Figure 8: Scatterplot of Total remaining cycle time predictions and actual values, with the total remaining cycle time on the Y-axis and the time spent on the X-axis.

Figure 9 shows the actual remaining cycle time and the predicted total remaining cycle time, as predicted by the average predictor and the non-parametric predictor based on only data-attribute variables. in Figure 9 the cycle time since the beginning of a case is shown on the horizontal axis, and the actual or predicted remaining cycle time on the vertical axis. Figure 10 show the mean square error of these predictors, in which the horizontal axis shows the time since the beginning of a case and the vertical axis shows the mean square error.

As shown in Figure 8, each prediction or measurement is a single point, the lines in the charts in this chapter are based on histograms. A histogram works as follows; the domain of the horizontal axis (or X-axis) is divided into bars. For these charts we have chosen a bar to be 7 days wide, as the longest case is 252 days long we will have 36 bars. Each prediction or measurement falls into a bar depending on its X-value, both the X-values and the Y-values(vertical axis) of all points in the bar are averaged. The average X-value

and the average Y-value become the coordinates for the single point of that bar. This way we construct a single line based upon a great deal of measurements. As the line represents the average value, we add the 95% confidence interval to the chart in the form of the coloured area surrounding the line. Note that we must assume a normally distributed error in order to calculate the confidence interval, which does not influence the predictions or the prediction method.

In Figure 9 on page 45, the line representing the average cycle time predictor should be a straight line with a constant variance. The reason this should be a straight line is because it is the prediction based on the average cycle time minus the time since the beginning of the case. Because the chart shows the predicted remaining cycle time against the time since the beginning of the case this becomes a straight line. The reason this is not a completely constant straight line (i.e. no variance) is because the average cycle time prediction is (just like the non-parametric predictor) "trained" on 90% of the log and tested on 10%, then repeated 10 times on a different stretch, resulting in a prediction for each case in the log. This 90% is a different part of the log each repetition, thus resulting in a different average each repetition, therefore the line is shown is the average of 10 slightly different lines.

Discussion of the total remaining cycle time prediction

Table 6: Results Total Remaining Cycle Time Prediction

Predictor	**Variables used**	**Mean Square Error** ± 95% confidence interval	
Average Predictor	-	2687.47	± 133.69
Nonparametric Predictor	only activity durations	5134.12	± 320.02
Nonparametric Predictor	only activity occurrences	1754.61	± 115.38
Nonparametric Predictor	only case attributes	1532.85	± 103.25
Nonparametric Predictor	all	1573.42	± 106.44

Figure 10 shows that in the beginning and at the end of a process, the average predictor and the non-parametric predictor perform rather equal. In the beginning of a case, little information is available for each case, meaning all cases are very similar. The "local-average" is therefore more like the actual average. One can see in the chart that the error of the non-parametric estimator decreases after 10 days of the process. This shows that, as more information becomes available on each case, the prediction becomes better. In the last part of the chart it is visible that the average predictor continuously predicts 0 remaining cycle time, this is because these are the cases that take longer than the average. As the average prediction is the average total remaining cycle time minus the cycle time

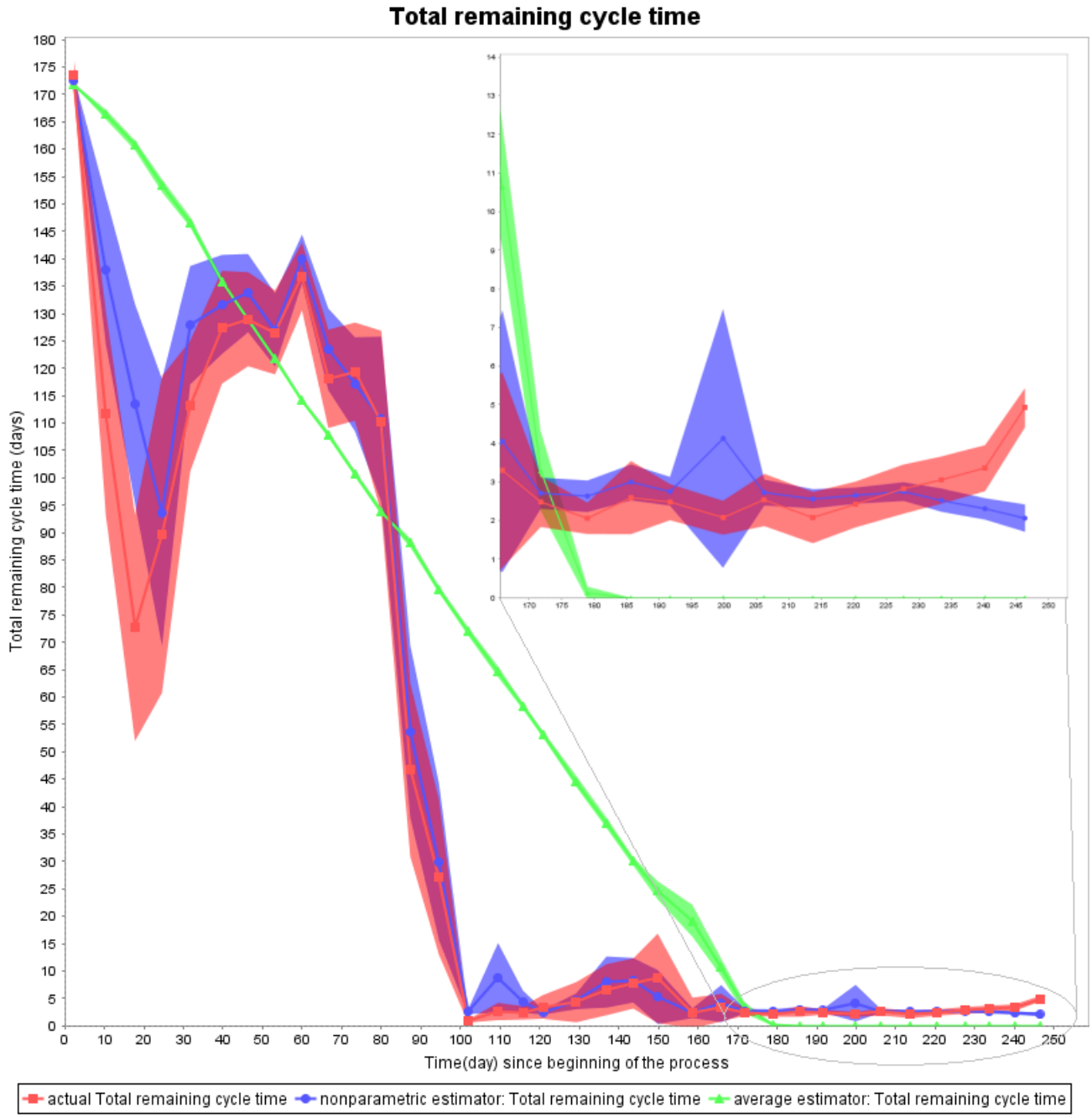


Figure 9: Total remaining cycle time prediction based on only data-attributes, with the total remaining cycle time on the Y-axis and the time spent on the X-axis. Each line is based on a histogram, and the coloured area surrounding it is the 95% confidence interval

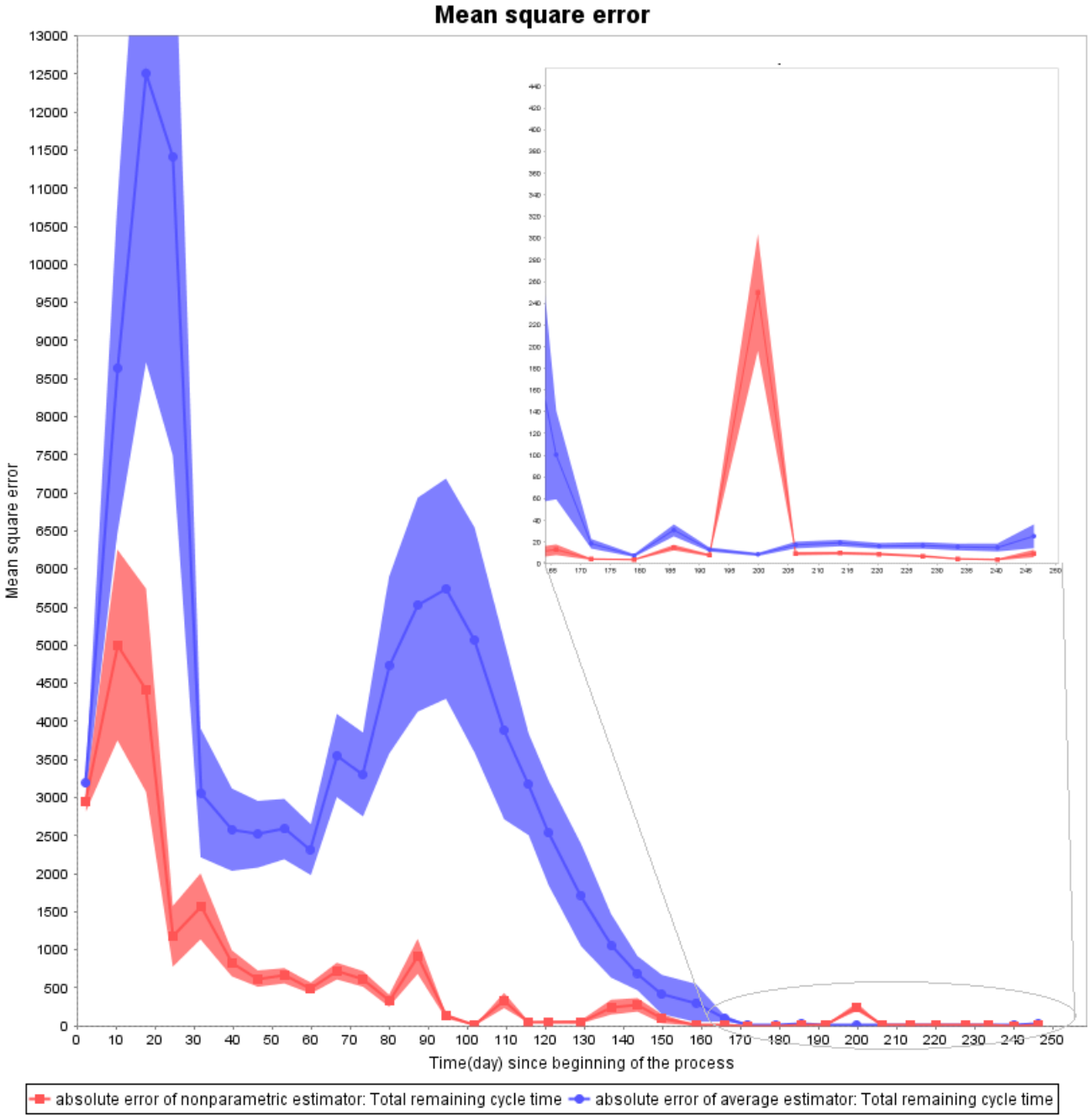


Figure 10: Mean square error of Total remaining cycle time prediction based on only data-attributes, with the mean square error on the Y-axis and the time spent on the X-axis. Each line is based on a histogram, and the coloured area surrounding it is the 95% confidence interval

so-far, this results in a negative value. As it is impossible to have a negative remaining cycle time, 0 is predicted.

Durations Table 6 shows that the durations are the worst basis for a prediction. This is explained by the fact that the durations in the log were not recorded such that they are an accurate representation of the duration of activities in the process. Some activities have durations that correspond to the actual duration of those activities, while other durations correspond to the time it took to input the results of the activities into the computer. As a result, the durations of activities present hardly any information for the non-parametric predictor to compare cases such that it can accurately predict the remaining cycle time. Because the durations of activities present so little predictive value, the nonparametric predictor using durations performs worse than the average based predictor.

Attributes The attribute based predictor performs the best, which is not surprising as the attributes contain information on the case. Information detailing the difficulty of a case, or other information that has a direct relation to the duration of that case. The bandwidth values (shown in Appendix D on page xvii) for the non-parametric predictor show that the attributes "new_queue", "id", and "priority" are the most influential attributes. The "priority" attribute is either "True" or "False", the name suggests that this attribute indeed has a big influence on the process time of a case. The attribute "new_queue" appears to be the next activity for this case, therefore this attribute changes at each activity. Another attribute that is relatively important is "queue", which indicates the activity that was just completed. As the non-parametric predictor uses all variables to compare and selects the most relevant cases, the combination of "queue" and "new_queue" are a good basis for comparison. The fact that the attribute "id" is an important factor in the prediction is interesting. An attribute that is much like "id", is the attribute "name". Both attributes are unique identifiers for cases. As the cross-validation splits up the log, it ensures that the measurement of a case are never used to make a prediction for that same case. Therefore, there is no prediction for which this attribute will be equal. Because the bandwidth is low, and there is not a single prediction for which this attribute is equal, it simply becomes a scalar for the weights and it is therefore irrelevant to the prediction. However, this shows that it is very difficult to derive information from the values in the bandwidth, as there are many factors to take into account.

The last part of the charts (starting around 100 days) are the predictions for the cases that take the longest. However, the (average) actual remaining cycle for these cases is relatively constant. This suggests that one of the last activities has an exponentially distributed cycle time for these cases. We will come back to this with the prediction of the remaining cycle time of the occurrence of the activity "0Z12 Hertaxeren".

Occurrences The results in Table 6 shows that the occurrence based predictor has an accuracy not much lower than the attribute based predictor. This means that the sequence of events for a case is almost equally useful in the prediction of the duration as information on the case itself. Apparently there is a sequence of activities that indicate a degree of complexity, thus enabling the non-parametric predictor to estimate a longer duration. In Figure 58 in Appendix C, a Petri-net is shown with a colour in each place. The colors (blue, yellow and red) indicate a degree of time that is spent in that place on average. An example of a trace with a very long process time (on average) is; "Domain:heus1", "0Z02 Voorbereiding", "0Z06 Stop Vordering", "0Z08 Beoordelen", "0Z12 Hertaxeren", "0Z16 Uitspraak", '0Z20 Administratie" and "0Z24 Start Vordering". An example of a very short trace is "Domain:heus1" followed by "0Z04 Incompleet". Many different traces are possible in this model, each with different average process times. The non-parametric regression compares cases based on the activities that occurred, it is therefore inevitable that the predictions for cases with an equal sequence are equal.

All predictor variables Comparing the figures in Table 6 shows that the mean square error of the combined predictor of the occurrences, the attributes and the durations is only slightly higher than the mean square error of the predictor based only on attributes. Although the expectation was that the addition of various less relevant predictor variables decreases the accuracy, the accuracy suffered only slightly. This result shows that the nonparametric predictor is capable of selecting the best predictor variables.

Comparison In comparison, the predictor with data-attributes performs best. This is not surprising as the data-attributes contain information such as the priority of the case, the difficulty of the case and some information on recent and future activities. The predictor based on occurrences is also a good predictor, this indicates that a better prediction can be made without having to store privacy sensitive information.

The computation of the bandwidth selection is very slow, which was expected to be a problem as it is an exponential computation (see Section 3.6). The number of cases in the experiments in this example is not extraordinary large, 706 cases is probably a small number for large companies. The computation for the bandwidths took less than a day for both the attributes-only and the occurrences-only predictions, while the bandwidth-selection for the predictions with a combination of all variables took almost three days. Fortunately, for the practical application of the approach the bandwidth only needs to be calculated once, after it has been found, the predictions themselves are computed in linear time. This leads to a suggestion for future work; the idle-time bandwidth optimization. As the computers providing predictions are usually not busy during the nights and weekends, they can continue optimizing the bandwidth. More suggestions for future work are found in Chapter 7.

5.1.3 Remaining cycle time until "0Z12 Hertaxeren" prediction

Results of the remaining cycle rime until "0Z12 Hertaxeren" Prediction Similar to the Total remaining cycle time, we have tested the Remaining cycle time until "0Z12 Hertaxeren". As explained in Section 4.5 for the prediction of the remaining cycle time until a specific activity, we assume that this activity will in fact occur. In a real life application of this method, we assume the occurrence is first predicted after which the remaining cycle time is predicted only when the activity is predicted to occur. Therefore we limit the predictions to cases in which "0Z12 Hertaxeren" actually occurs.

Table 7: Results Remaining Cycle Time until "0Z12 Hertaxeren" Prediction

Predictor	**Variables used**	**Mean Square Error** ± 95% confidence interval	
Average Predictor	-	399.54	± 38.05
Nonparametric Predictor	only activity durations	1052.32	± 38.05
Nonparametric Predictor	only activity occurrences	307.05	± 46.14
Nonparametric Predictor	only case attributes	257.34	± 33.23
Nonparametric Predictor	all	385.67	± 87.44

Because the average time until "0Z12 Hertaxeren" is shorter than the average time until the end of the process, the absolute errors are also smaller. The average predictor shows that on average the activity "0Z12 Hertaxeren" occurs after 207 days, while the average time until the end of the process was 174 days. This shows that cases in which the activity '0Z12 Hertaxeren" does not occur are cases that take relatively little time. As a result, the mean square error is relatively low.

Discussion of the remaining cycle time until "0Z12 Hertaxeren" Prediction

Durations The durations are by far the worst predictor for predicting the remaining cycle time until the activity "0Z12 Hertaxeren", the results in Table 7 show that the mean square error of this predictor is significantly larger. Again, the durations of activities have very little predictive value in this process.

Attributes Similar to the prediction of the total remaining cycle time, the predictor based on attributes performs the best. Figures 11 and 12 show the estimations and mean square errors of the nonparametric predictor using attributes. The charts show how the average prediction is the best possible estimation at the start of the process, these charts also show how the nonparametric predictor mimics the average predictor in the beginning.

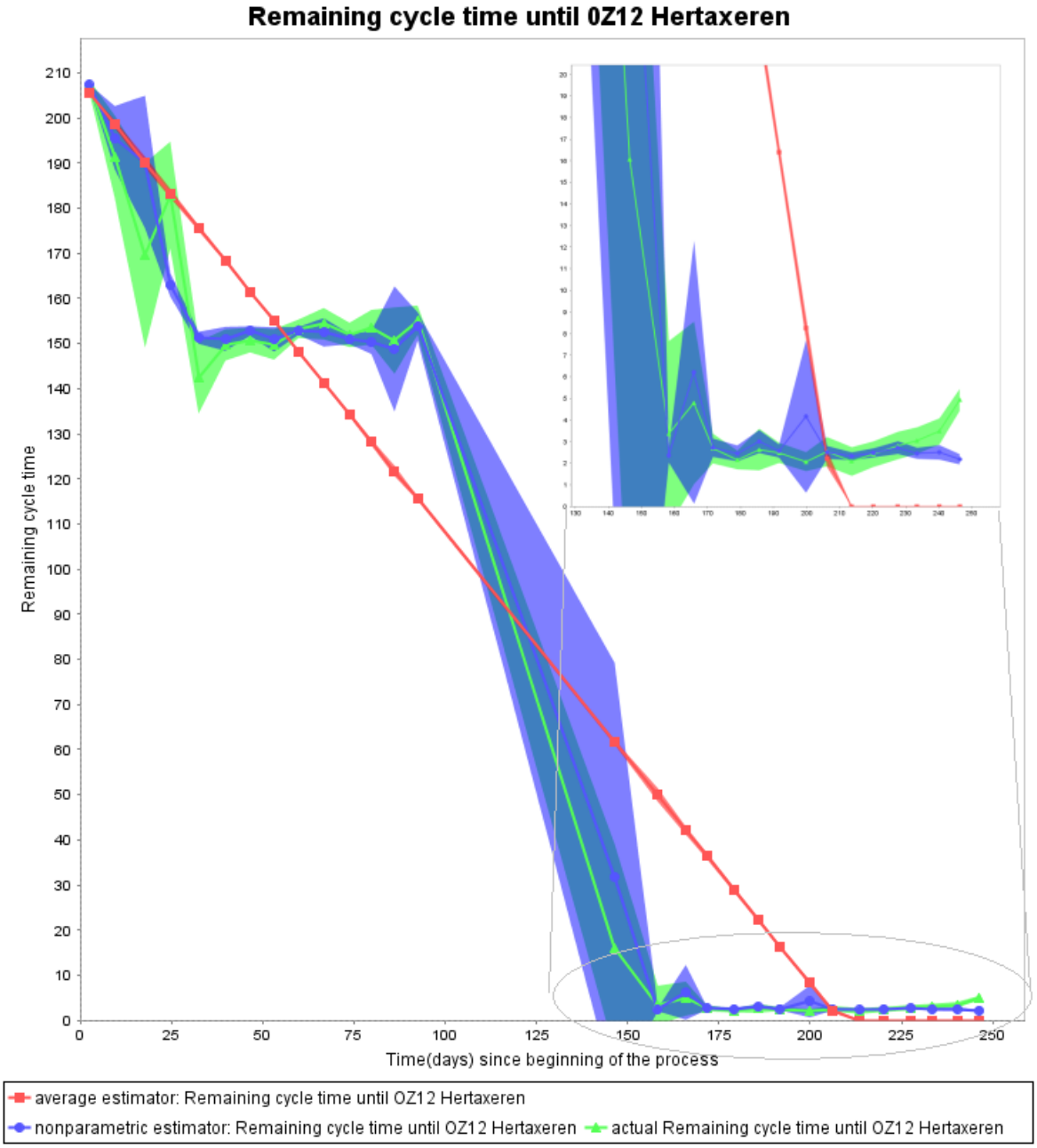


Figure 11: Remaining cycle time until "0Z12 Hertaxeren" predictions with only data-attributes

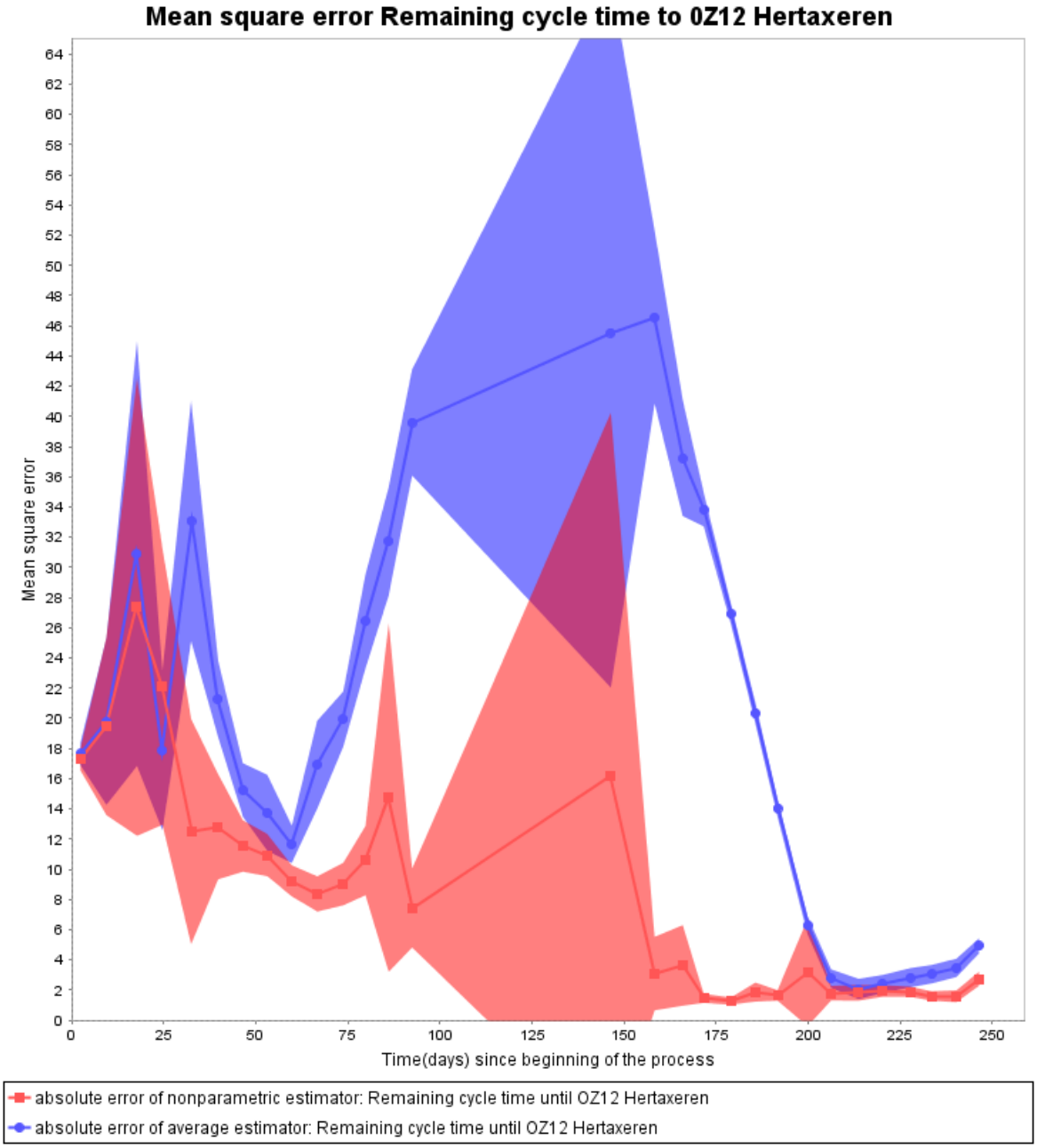


Figure 12: MSE of Remaining cycle time until "0Z12 Hertaxeren" predictions with only data-attributes

As we noticed in the discussion of the results for the prediction of the total remaining cycle time, the average remaining cycle time stays constant which suggests there are most likely activities that have an exponentially distributed cycle time. Because the predictions of "0Z12 Hertaxeren" also shows this, it is safe to say that the sequence of activities that lead to "0Z12 Hertaxeren" contains the activity that is exponentially distributed.

Occurrences The occurrences based predictions are not the best, but certainly better than the all most predictions, as it is only outperformed by the nonparametric predictor using attributes. The bandwidths of occurrence based prediction of the remaining cycle time until "0Z12 Hertaxeren" shows that the occurrences of the activities "0Z08 Beoordelen", '0Z04 Incompleet", "0Z10 Horen", and "0Z15 Zelf Uitspraak" are important. The process model of the case study in Appendix C shows that all sequences that include the activities "0Z10 Horen" and "0Z15 Zelf Uitspraak" are not likely to include "0Z12 Hertaxeren", while sequences that include "0Z08 Beoordelen" are likely to also include "0Z12 Hertaxeren". Comparing cases on these activities provides a good performance.

All predictor variables The combination of all predictor variables provides less accuracy compared to the occurrences or the attributes, however more than the accuracy of the durations. This shows that more predictor variables can decrease the performance of the nonparametric predictor, because then there are also more irrelevant predictor variables. This leads to a suggestion for future work, the removal of redundant predictor variables.

Comparison The accuracy of the nonparametric predictor compared to the average predictor is not significantly higher, nonetheless the results show that the nonparametric predictor performs better in this example. This example also shows that the presence of irrelevant predictor variables decreases the accuracy of the predictor.

5.1.4 Prediction of the occurrence of "0Z12 Hertaxeren"

Results of the Prediction of the occurrence of "0Z12 Hertaxeren"

Using the same settings (10-fold cross-validation, etc) as the remaining cycle time prediction, we have tested the predictions of the occurrences. We must first note that there are some limitations to the software used for this; as the package used for the regression does not allow us to adapt the bandwidth selection functions easily, we could only round the predictions themselves to natural numbers. Internally the prediction method calculates an optimal bandwidth for a continuous response value, the bandwidth is therefore not optimized for the rounded predictions but for the unrounded predictions. This could lead to a suboptimal bandwidth.

Table 8: Definition of results

example		**Actual**	
		1	**0**
Predicted	**0**	A	B
	1	C	D
False Positive Rate		$\frac{D}{D+B}$	
False Negative Rate		$\frac{A}{A+C}$	
Total Error		$\frac{D+A}{A+B+C+D} \times 100\%$	

In this experiment each prediction can only be 0 or 1, therefore the mean square error is not a good predictor in this case. A different method, using the definition of Type I and Type II errors, is to show the False Positive Rate and the False Negative Rate. The False Positive Rate shows the proportion of cases with 0 occurrences that were erroneously predicted to have 1 occurrence, equally the False Negative Rate shows the proportion of cases with 1 occurrence that were erroneously predicted to have 0 occurrences. Table 8 shows an overview of the definitions of all results shown. In this table the relation between the False Negative Rate, the False Positive Rate and the tables with predictions versus actual values are shown. Table 9 shows an overview of the accuracy of the results. Tables 10 and 11 show the results of all predictions, the tables show the number of predictions categorised by two dimensions, actual value and predicted value.

Recall that the average predictor predicts the average number of occurrences. For this experiment it predicted 1 occurrence, i.e. it predicted that the activity "0Z12 Hertaxeren"

Table 9: Overview Results "0Z12 Hertaxeren" Prediction

Predictor	**Variables used**	**False Negative Rate**	**False Positive Rate**
Average Predictor	-	0	1
Nonparametric Predictor	only activity durations	0.761740	0.229557
Nonparametric Predictor	only activity occurrences	0.017265	0.510878
Nonparametric Predictor	only case attributes	0.001727	0.933983
Nonparametric Predictor	all	0.099102	0.603151
Predictor	**Variables used**	**Total error**	
Average Predictor	-	31.5%	
Nonparametric Predictor	only activity durations	59.4%	
Nonparametric Predictor	only activity occurrences	17.3%	
Nonparametric Predictor	only case attributes	29.6%	
Nonparametric Predictor	all	25.8%	

Table 10: Results "0Z12 Hertaxeren" prediction using the non-parametric predictor

nonpar. predictor case attributes		**Actual**	
		1	**0**
Predicted	**0**	5	88
	1	2891	1245

nonpar. predictor occurrences		**Actual**	
		1	**0**
Predicted	**0**	50	652
	1	2846	681

nonpar. predictor durations		**Actual**	
		1	**0**
Predicted	**0**	2206	1027
	1	690	306

nonpar. predictor all		**Actual**	
		1	**0**
Predicted	**0**	287	529
	1	2609	804

Table 11: Results "0Z12 Hertaxeren" Prediction using the average occurrence predictor

avg. predictor		**Actual**	
		1	**0**
Predicted	**0**	0	0
	1	2896	1333

will always occur. As a result the total error of the average predictor will always be lower than 50%, in this case the error was 31.5%.

Discussion of the prediction of the occurrence of "0Z12 Hertaxeren"

Durations Similar to the other predictions, the durations provide little insight into the process as the predictions based on durations perform the worst.

Attributes Table 9 shows that the predictor based on attributes is second best if we look at the Total error. However, the False Positive/Negative Rate are more interesting. The False Positive Rate shows the rate of cases for which we (erroneously) predicted that "0Z12 Hertaxeren" would occur, and the False Negative Rate is the rate of cases for which we erroneously predicted "0Z12 Hertaxeren" would not occur. For this predictor the False Positive Rate is almost 1 and the False Negative Rate is almost 0. The results in Table 10 show that this is because almost all predictions were predicted to have an occurrence of "0Z12 Hertaxeren".

Appendix F on page xix shows that the attributes "error", "name", "id", and "priority" are the most influential variables. Judging by their names both "error" and "priority"

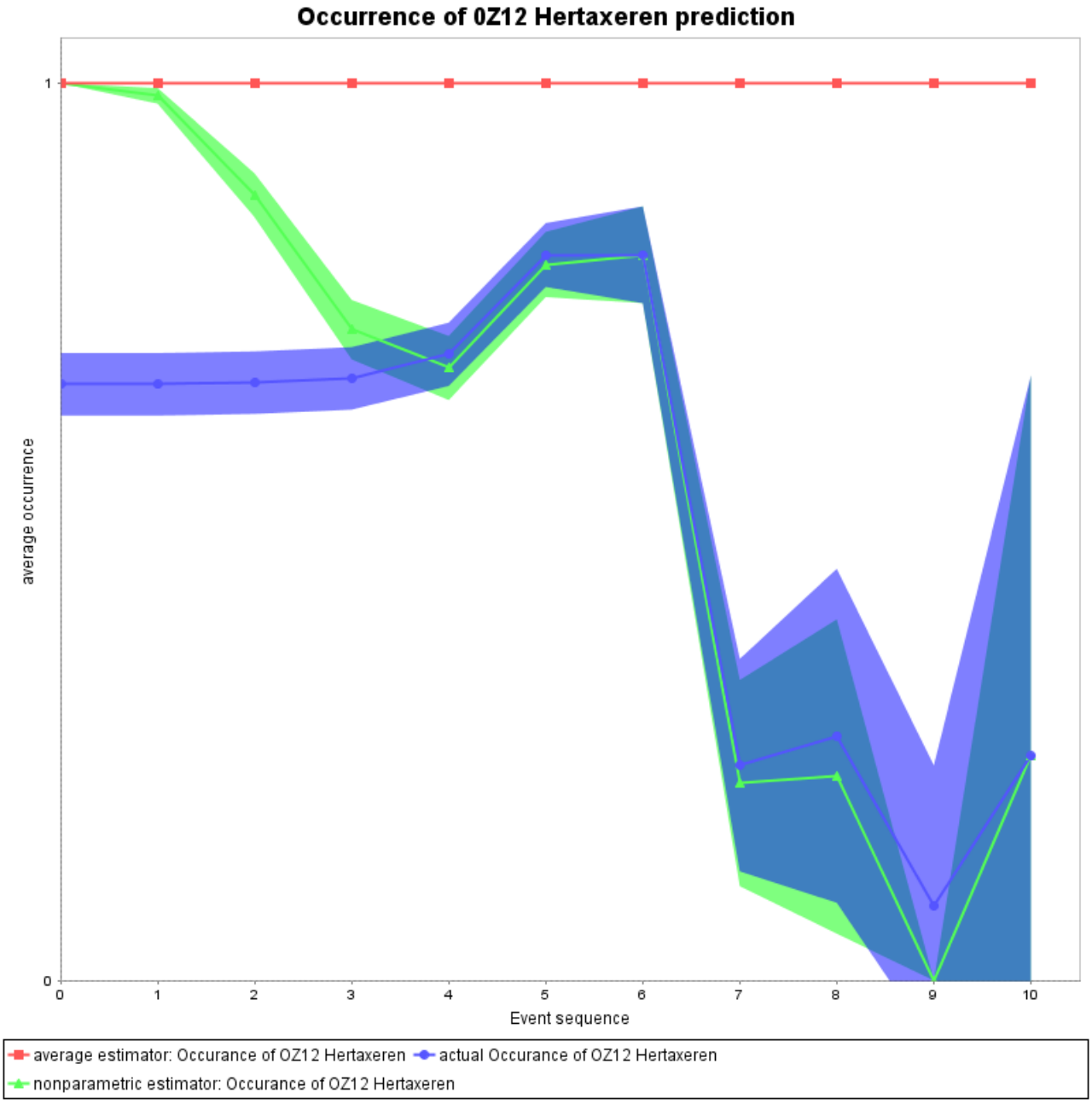


Figure 13: Prediction of the occurrence of "0Z12 Hertaxeren" with only occurrenes

are obviously important predictor variable. As with the total remaining cycle time, the other two attributes "name" and "id" are always unique and therefore always irrelevant variables.

Occurrences The non-parametric predictor based on occurrences performed the best, with an error of 17%. The occurrences present information on what activities have already occurred. By looking at the process model in Appendix C on page xv and the bandwidth values in Appendix E on page xvii, we can determine which activities are most influential in the prediction. The occurrences "OZ04 Incompleet" is one of the occurrences with a low bandwidth value, the process model indicates that this is indeed a very short path that does not lead to "0Z12 Hertaxeren". The path "0Z02 Voorbereiden", "0Z09 Wacht Beoordeling" is another path that indicates "0Z12 Hertaxeren" will not occur. However, the path "0Z02 Voorbereiden", "0Z12 Hertaxeren" is a path that always leads to "0Z12 Hertaxeren". Although this is logical for a human, the non-parametric regression does not know what activity it is predicting. The non-parametric regression only knows that it has a set of measurements with predictor variables and a single response variable, on which it must base a prediction for a new case.

All predictor variables The non-parametric predictor using all predictor variables performs better than the predictor using attributes and worse than the predictor using occurrences. As with the remaining cycle time until "0Z12 Hertaxeren" this shows that the addition of more predictor variables can decrease accuracy. However, the accuracy of the non-parametric predictor using all predictor variables is much better than the accuracy of the predictor using durations, therefore this also shows that the non-parametric predictor is capable of selecting the better predictor variables, but still suffers in accuracy.

Comparison In comparison, the non-parametric predictor using occurrences has by far the highest accuracy, which shows that the non-parametric predictor is in fact better than the average predictor. The prediction of the occurrence of "0Z12 Hertaxeren" also shows that the non-parametric predictor is capable of selecting the most relevant predictor variables, but the non-parametric predictor still suffers in accuracy if many irrelevant predictor variables are present.

5.1.5 Conclusion of the case study

For the case study we applied the non-parametric prediction method to a log taken from practice. We have tested all three types of prediction, namely the total remaining cycle time, the remaining cycle time until an activity and the occurrence of an activity. The results of the case study show that the non-parametric predictor is substantially better given the right selection of variables. However, the non-parametric predictor is not significantly

better than the average predictor in all types of predictions. Furthermore the case study shows that, although the non-parametric predictor suffers in accuracy from irrelevant predictor variables, it is capable of selecting the relevant predictor variables from irrelevant predictor variables.

In order to explain the behaviour of the non-parametric predictor in specific types of processes, the rest of this chapter shows simulations.

5.2 Simulations

By testing the prediction method on a log created by a simulated processes, it is possible to verify the output. In other words, we can formulate a hypothesis about the predictions for a simulated process, the results either confirm or reject the hypothesis. All processes shown in this chapter, are simulated using CPN-Tools [10]. CPN-Tools is a tool for simulating colored Petri Nets, the tool is maintained by the CPN group of the University of Aarhus in Denmark. All CPN-Tools models are shown in Appendix B on page viii, for each simulation experiment 1000 cases were used. The number of 1000 cases is chosen for practical reasons, if there are more cases the time needed to compute the predictions on each simulation becomes too long. (For 1000 cases some of the computations already took 3 days.)

5.2.1 Comparison of the non-parametric and the average predictor

The previous chapters described how the non-parametric predictor bases its predictions on a local average of cases that most resembles the case for which we predict. In order to compare cases, any combination of activity occurrences, the activity durations, and/or data-attributes is used. The average predictor relies on the total average, and the time-spent-so-far for predictions. If there are no good predictor variables to compare cases, the non-paramatric predictor based on occurrences will expand the the local average such that it constructs an average prediction for each possible prefix. Therefore, the non-parametric predictor using the optimal predictor variables is always similar or better than the average predictor, even in processes in which the average prediction is the best possible prediction.

Hypothesis 1. *The mean square error of the non-parametric predictor is not significantly higher than the mean square error of the average predictor.*

In order to test this hypothesis we look at two different aspects of a process, the cycle time distribution and the structure of the process. Both the cycle time distribution and the structure of the process influence the performance of a predictor. Two simulations are constructed, each constructed to test one of these two aspects. The first process we simulate is a process where all cycle times are exponentially distributed. In this process the cycle times of completed activities have no predictive value, also the average is the

best prediction for the remaining cycle time. The second simulated process is a parallel process, in which the sequence of activities is unpredictable. In both simulations the worst-case-scenario are tested, if both simulations show the non-parametric predictor is not significantly higher than the mean square error of the average predictor the hypothesis is confirmed.

Simulation 1: cycle time distribution Figure 14 shows a sequential process with exponentially distributed process times. This process is simulated by a CPN-Tools simulation, shown in Appendix B Figure 50. Because the process times are independently distributed, the actual remaining cycle time is independent of the value of the duration of any previous activity. The average predictor however, is dependent on the value of the duration of activities A, B, and C. Whereas, the non-parametric predictor compares cases and bases a prediction on the remaining cycle time of the most similar cases. In the case of the non-parametric predictor using occurrences this results in a average for each possible prefix, which means an average is constructed at each point in the process. The non-parametric predictor based on durations does the same, although the actual value of the duration of an activity is irrelevant in this process, any non-zero duration is equal to an occurrence of at least 1. This means the non-parametric predictor constructs a prediction based on the average remaining cycle time at each point in the process instead of an average remaining cycle time based on the time spent so far. Therefore, this simulation is supposed to show that the mean square error of the non-parametric predictor is not significantly higher than the mean square error of the average predictor. The process is

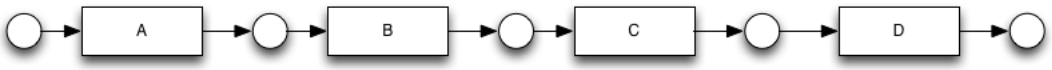


Figure 14: Simulated Process 1

simulated both with and without queueing, to provide an extra insight into the effect of queueing on the performance of the predictors. The activities are M/M/1-queueing models for which the occupation-rate is varied to be 0, 0.7 and 0.9. We will denote Simulation 1a as the simulation of a sequential process without queues, Simulation 1b as the simulation of a sequential process with queues and occupation-rate 0.7, and Simulation 1c as the simulation of a sequential process with queues and occupation-rate 0.9. Figure 51 in Appendix B shows the same sequential process with queues, in which each activity is modelled by Figure 52. In order to vary the occupation rate, we have kept the (exponentially distributed) arrival rate steady at an average of 10 minutes between each arrival. For Simulation 1b the average process time of each activity was 7 minutes, in Simulation 1c it was 9 minutes. Because Simulation 1a is done without queues, there was no arrival rate and the average process time of each activity was 10 minutes.

Table 12: Results Simulation 1a : sequential process no queues

Predictor	**Variables used**	**Mean Square Error** ± 95% confidence interval	
Average Predictor	-	298.59	± 22.95
Non-parametric Predictor	only activity durations	223.96	± 23.25
Non-parametric Predictor	only activity occurrences	204.11	± 22.58
Non-parametric Predictor	both	203.23	± 22.43

Table 13: Results Simulation 1b : sequential process with queues and occupation rate 0.7

Predictor	**Variables used**	**Mean Square Error** ± 95% confidence interval	
Average Predictor	-	2345.49	± 248.78
Non-parametric Predictor	only activity durations	2054.44	± 253.65
Non-parametric Predictor	only activity occurrences	1963.42	± 250.48
Non-parametric Predictor	both	1951.98	± 252.00

Table 14: Results Simulation 1c : sequential process with queues and occupation rate 0.9

Predictor	**Variables used**	**Mean Square Error** ± 95% confidence interval	
Average Predictor	-	17909.14	± 1927.06
Non-parametric Predictor	only activity durations	15576.97	± 1994.79
Non-parametric Predictor	only activity occurrences	15049.60	± 1939.57
Non-parametric Predictor	both	15097.09	± 1946.61

Results Simulation 1 Tables 12, 13, and 14 show the mean square errors of the simulations with their corresponding 95% confidence interval. Figures 15, 17,and 19 show histograms of the estimations of the non-parametric predictor with the lowest MSE, the average predictor and the actual predictions. Figures 16, 18, and 20 show charts with the mean square errors of the best performing non-parametric predictor and the average predictor. Figures 21, 23 and 25 show the scatterplots of the actual remaining cycle times, Figures 22, 24 and 26 show the scatterplots of the non-parametric predictor with the lowest MSE and the average predictor.

The hypothesis that the mean square error of the non-parametric predictor is not significantly lower than the mean square error of the average predictor, is confirmed. The results in tables 12, 13, and 14 show that in each simulation the non-parametric predictor

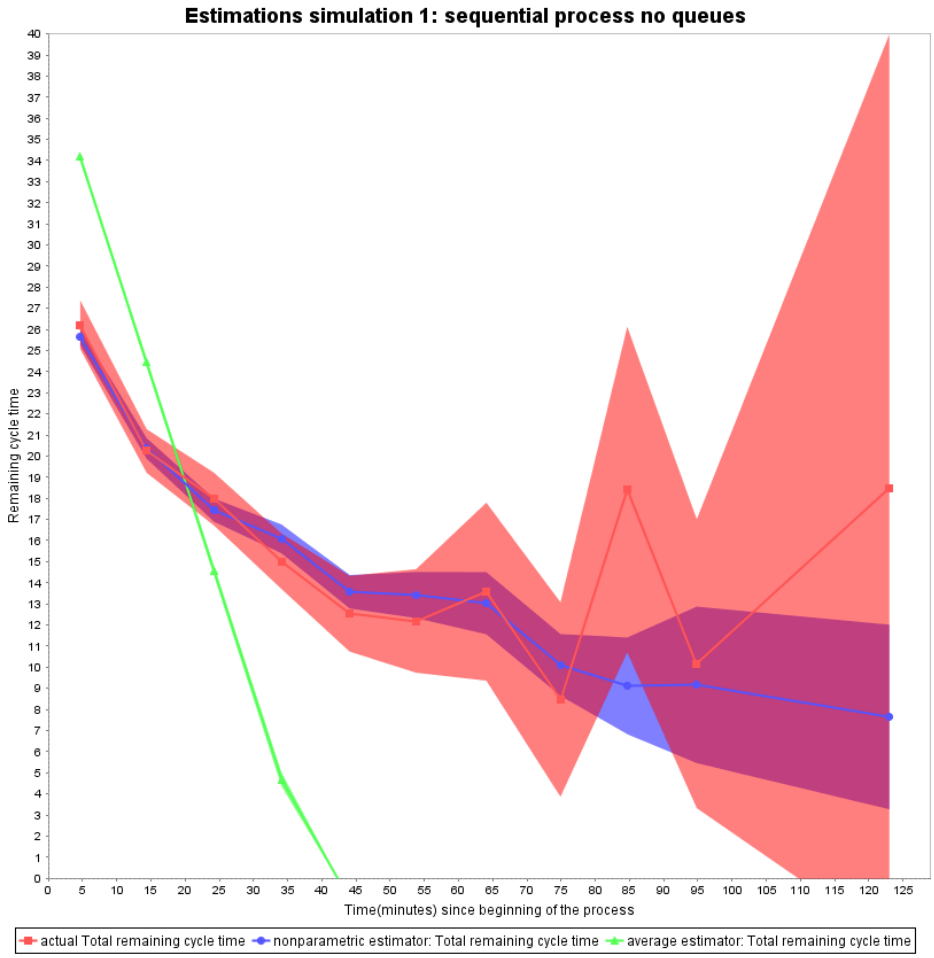


Figure 15: Total remaining cycle time - Simulation 1a: sequential process no queues

performed equally or significantly better. In Simulation 1a (sequential process without queues) the mean square error of the non-parametric predictor based on all predictor variables is significantly lower than the mean square error of the average predictor. In all other simulations the mean square error of the best-performing non-parametric predictor is lower but not significantly lower than the mean square error of the average predictor.

Discussion Simulation 1 In Figure 22, the line of the average predictor confirms that on average the total cycle time of a case is around 40 minutes. Figures 24 and 26 show that because of the queues in the process, the total average cycle time increases. More specifically it appears that with an occupation rate of 0.7 the average sojourn time[3] of the process is around 80 minutes, and with an occupation rate of 0.9 it is around 170

[3]sojourn time = process time + waiting time in the queue

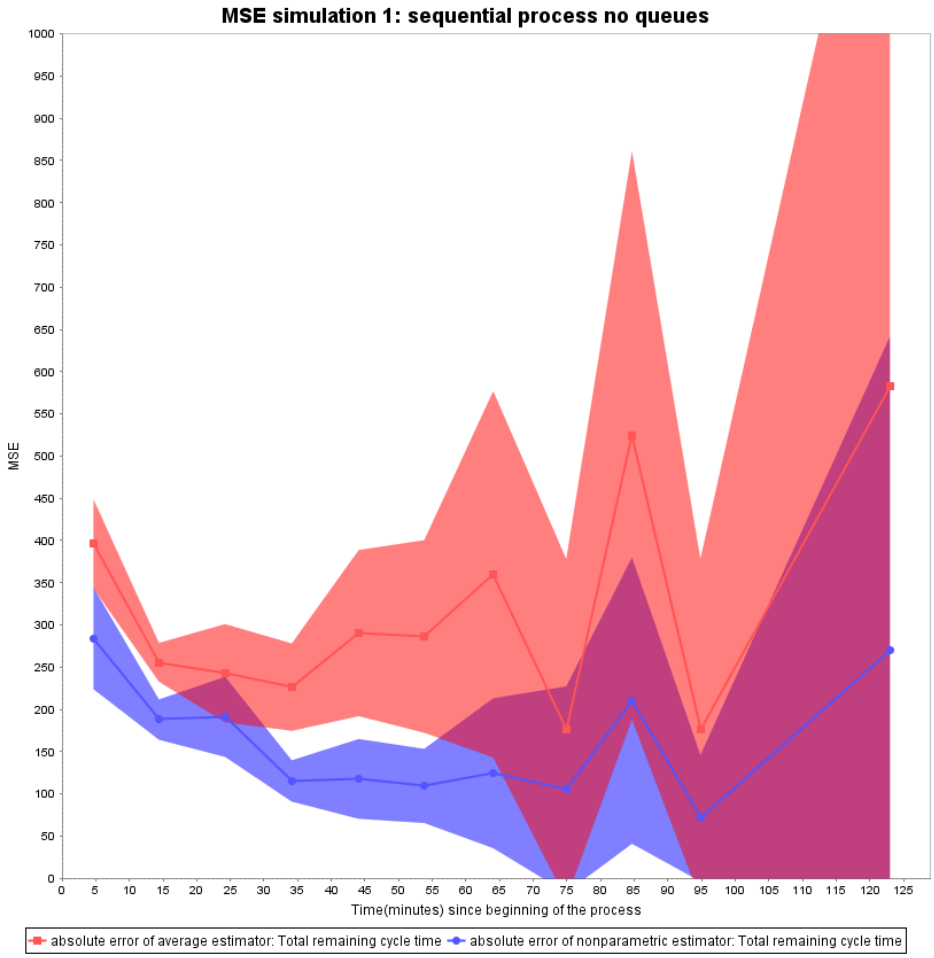


Figure 16: Mean Square Error - Simulation 1a: sequential process no queues

minutes. Note that because of the 10-fold cross validation, the average predictor consists of 10 different lines.

In all three simulations, the actual remaining cycle time fluctuates just above respectively 10, 20 and 50. This is because the remaining cycle time is dependent on the last activity which is exponentially distributed. However, what is more important is that this fits the average sojourn time for the complete process that we estimated in the previous paragraph. As these simulations are based on processes with $M/M/1$-queues, eventually the queues will become "stable", meaning that the queue-lengths will become stable and with the queue-lengths the average sojourn time. However, because these simulations are too short for the queues to become stable, the average remaining sojourn time will fluctuate more, which what we see in Figures 17 and 19. Because all four activities will have an equal arrival rate and process time, eventually all four activities will have an equal sojourn time. Therefore, if one activity has an sojourn time of about 20 minutes, the total sojourn time

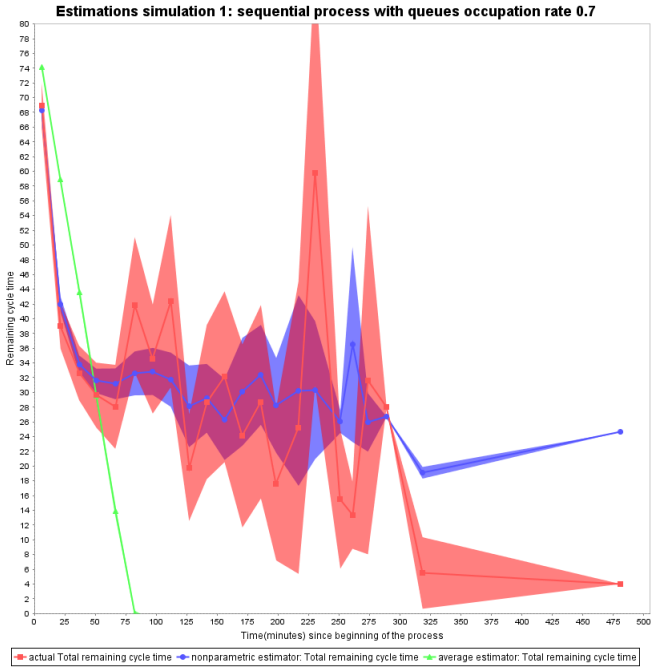


Figure 17: Total remaining cycle time - Simulation 1b: sequential process with queues and an occupation rate of 0.7

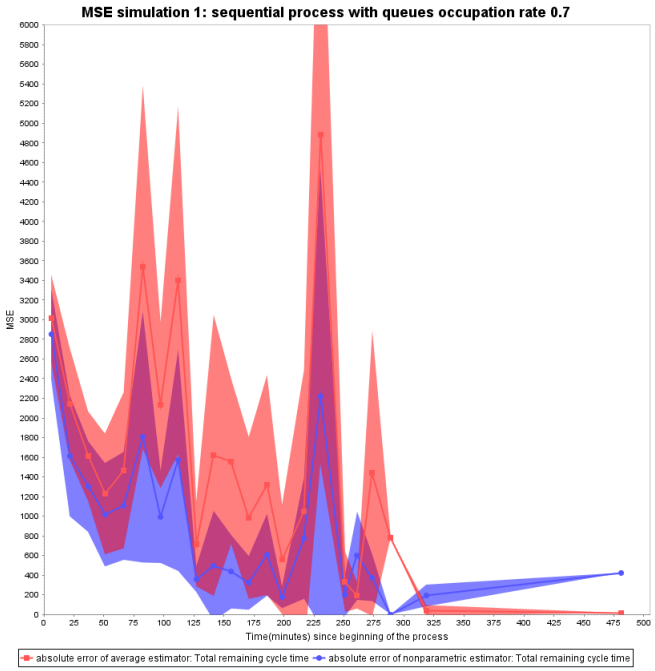


Figure 18: Mean Square Error - Simulation 1b: sequential process with queues and an occupation rate of 0.7

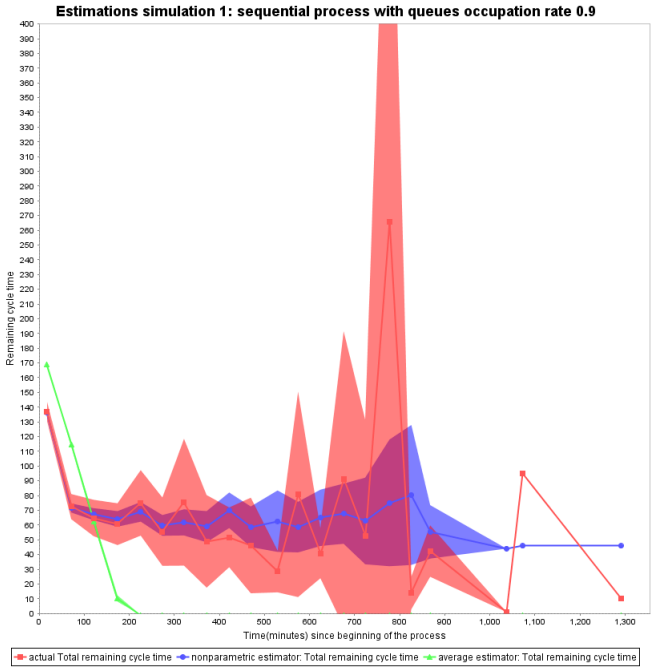


Figure 19: Total remaining cycle time - Simulation 1c: sequential process with queues and an occupation rate of 0.9

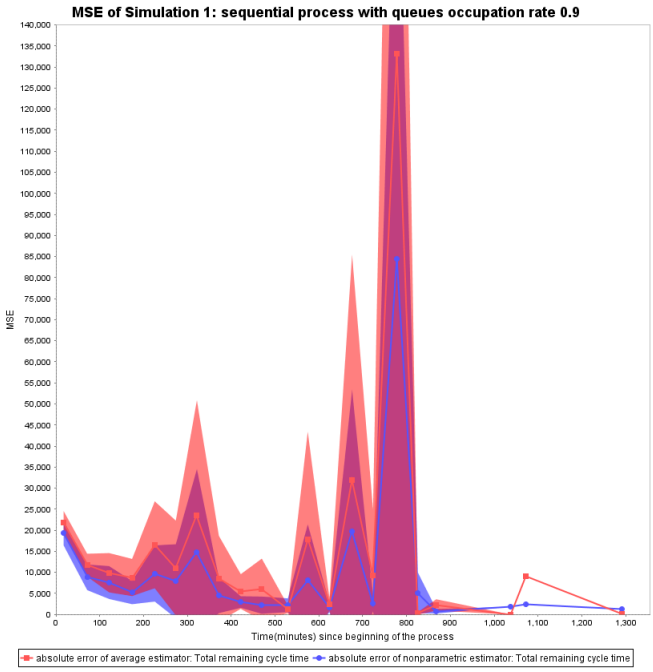


Figure 20: Mean Square Error - Simulation 1c: sequential process with queues and an occupation rate of 0.9

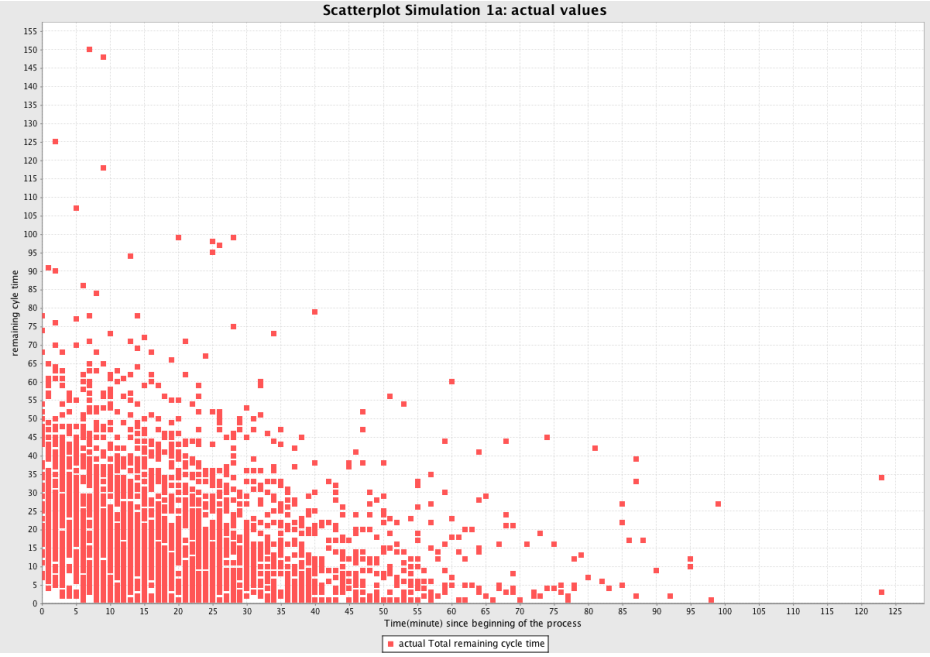


Figure 21: Scatterplot of total remaining cycle time - Simulation 1a: sequential process without queues

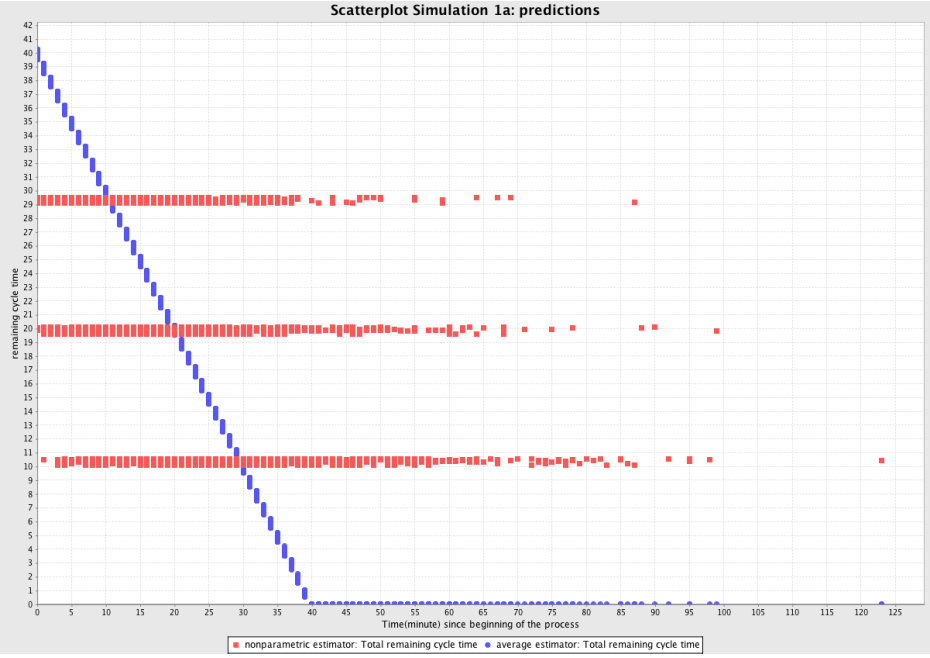


Figure 22: Scatterplot of predictions - Simulation 1a: sequential process without queues. The three red lines consists of the predictions for the three possible states in the sequential process.

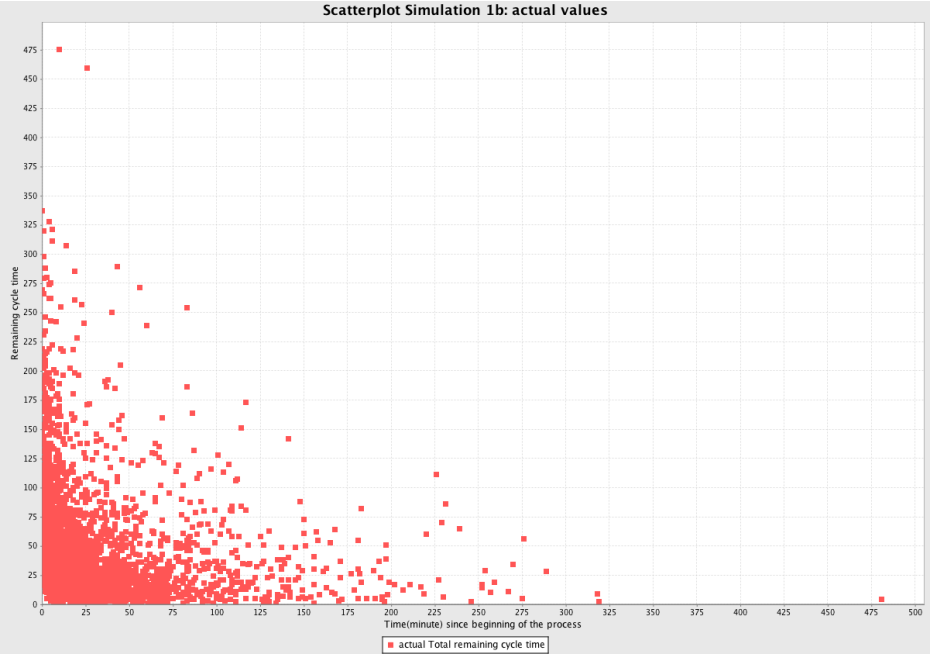


Figure 23: Scatterplot of total remaining cycle time - Simulation 1b: sequential process with queues and an occupation rate of 0.7

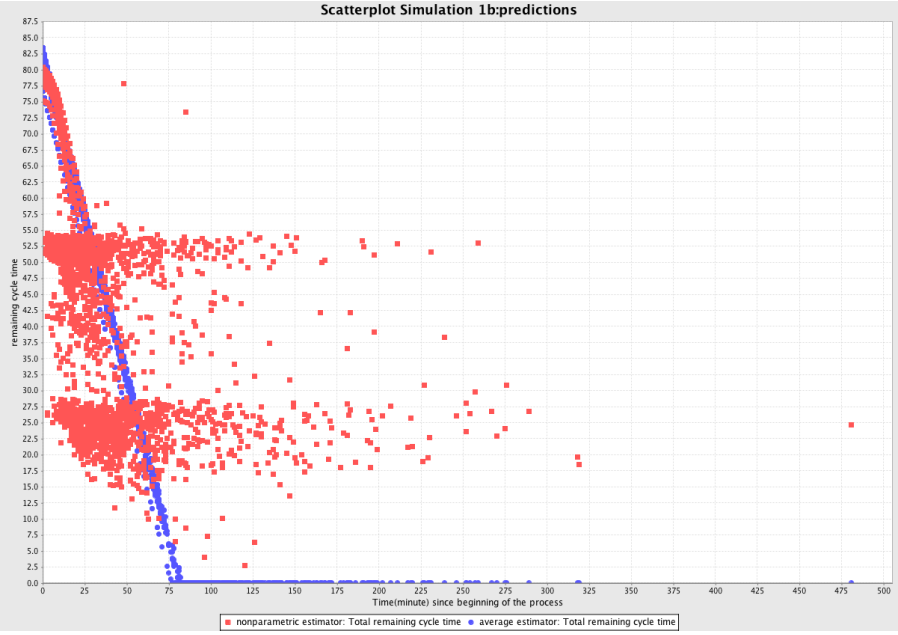


Figure 24: Scatterplot of predictions - Simulation 1b: sequential process with queues and an occupation rate of 0.7. Three clusters are visible, one for each state, but also dependent on the duration of each activity.

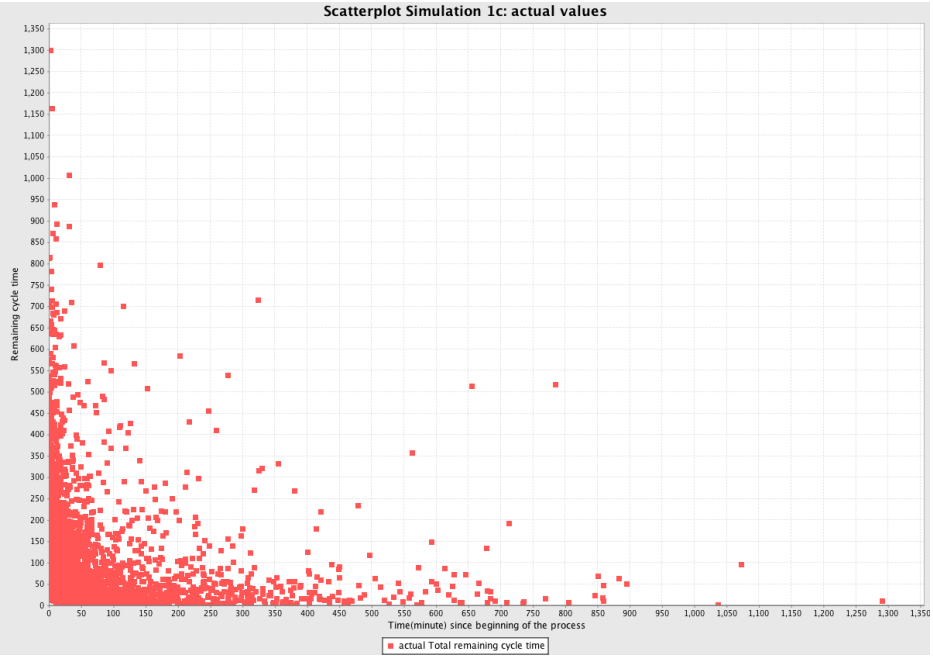


Figure 25: Scatterplot of total remaining cycle time - Simulation 1c: sequential process with queues and an occupation rate of 0.9

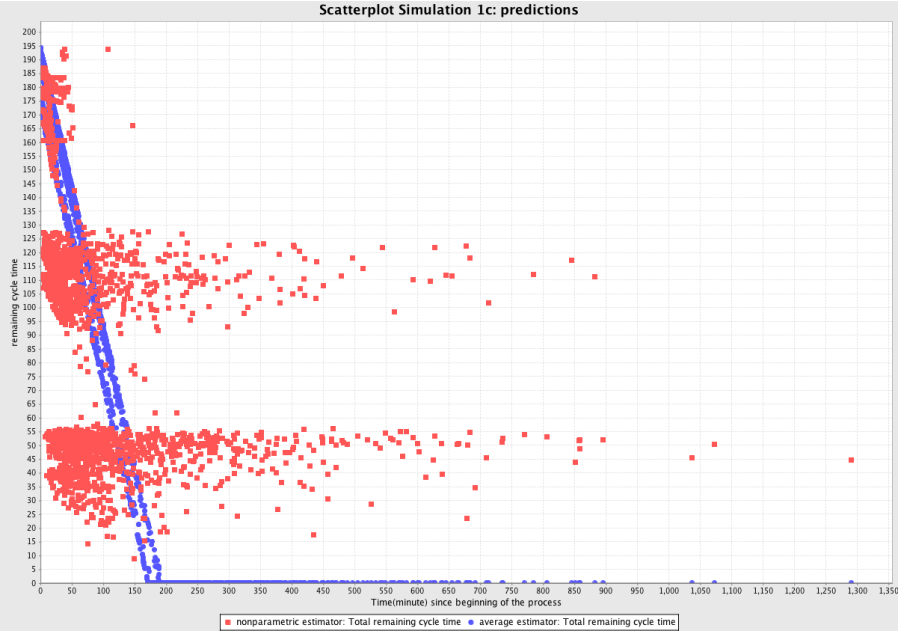


Figure 26: Scatterplot of predictions - Simulation 1c: sequential process with queues and an occupation rate of 0.9. Three clusters are visible, one for each state, but also dependent on the duration of each activity.

for this process will be around 80 minutes. This matches the average sojourn time found by the average predictor.

In Simulation 1a, the best performing non-paramatric predictor was the occurrence based, as is clearly shown in Figure 22 by the three distinct lines of predictions. As there are three different possible states (A, AB, and ABC), the non-parametric predictor has three local-averages resulting in three different predictions. This shows that the non-parametric predictor does not predict based on the actual values of the predictor variables, it compares cases and bases a prediction on the remaining cycle time as it was in those cases.

In simulations 1b and 1c, the non-parametric predictor performs best with a combination of durations and occurrences. As a result, the scatterplots of the predictions in figures 24 and 26 show that the predictions are more dependent on the time spent, and not just on the location in the process. There are still three different clusters visible that correspond to the three possible states, however, the first cluster appears to be based mostly on the duration of activity A, as it follows the line of the average predictor. This shows that, because of the simple nature of the non-parametric predictor, which is to compare cases, assign a weight and compute the weighted average, it allows for complex behavior. In this case it results in predictions that are partly dependent on the time spent and partly on the location in the process.

The non-parametric predictor has a lower mean square error than the average predictor in all three simulations. However as the difference is not significant, we can say that even in a process with exponentially distributed cycle times, the non-parametric predictor (with the best set of predictor variables) performs at least as good as the average predictor.

Simulation 2: process structure Figure 27 shows a parallel process, and the corresponding simulation model is found in Figure 53 in Appendix B. In this simulation activity A and activity D take no time, and activities $B1$, $B2$, $B3$, $C1$, $C2$, and $C3$ have independent normally distributed process times (> 0) with 10 days on average and 3 days standard deviation. In this process, neither the occurrences of activities nor the durations of activities have any predictive value. However, the time-spent-so-far does have predictive value. Therefore, this simulation tests if the mean square error of the non-parametric predictor is not significantly higher than the mean square error of the average predictor.

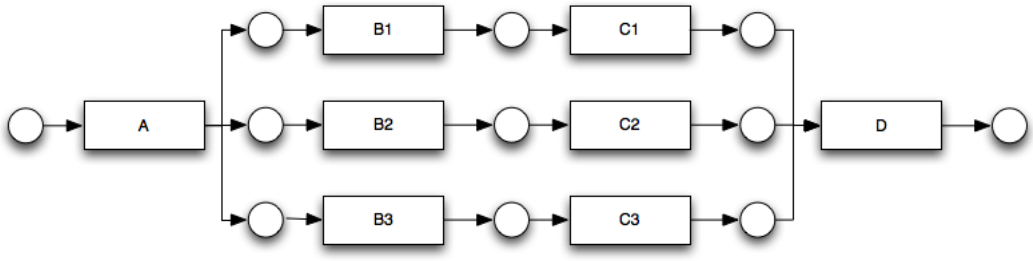


Figure 27: Simulated Process 2

Table 15: Results Simulation 2 : parallel process

Predictor	**Variables used**	**Mean Square Error** ± 95% confidence interval	
Average Predictor	-	5241.52	± 194.30
Non-parametric Predictor	only activity durations	4436.31	±176.40
Non-parametric Predictor	only activity occurrences	4988.29	± 194.43
Non-parametric Predictor	both	4317.40	± 170.28

Results Simulation 2 The results of the simulation are shown in Table 15, Figure 28 shows the estimations, and Figure 29 shows the mean square errors of the predictions. Furthermore, Figure 30 shows the scatterplot of the non-parametric predictions and Figure 31 shows the scatterplot with the actual values and the average predictions. The results show that the mean square error of the non-parametric predictor is significantly lower than the mean square error of the average predictor, therefore we confirm the hypothesis that the non-parametric predictor has similar or better performance.

Discussion Simulation 2 Figure 31 shows the scatterplot for the actual values and the average predictor, in this plot it can be seen that the parallel process is normally distributed

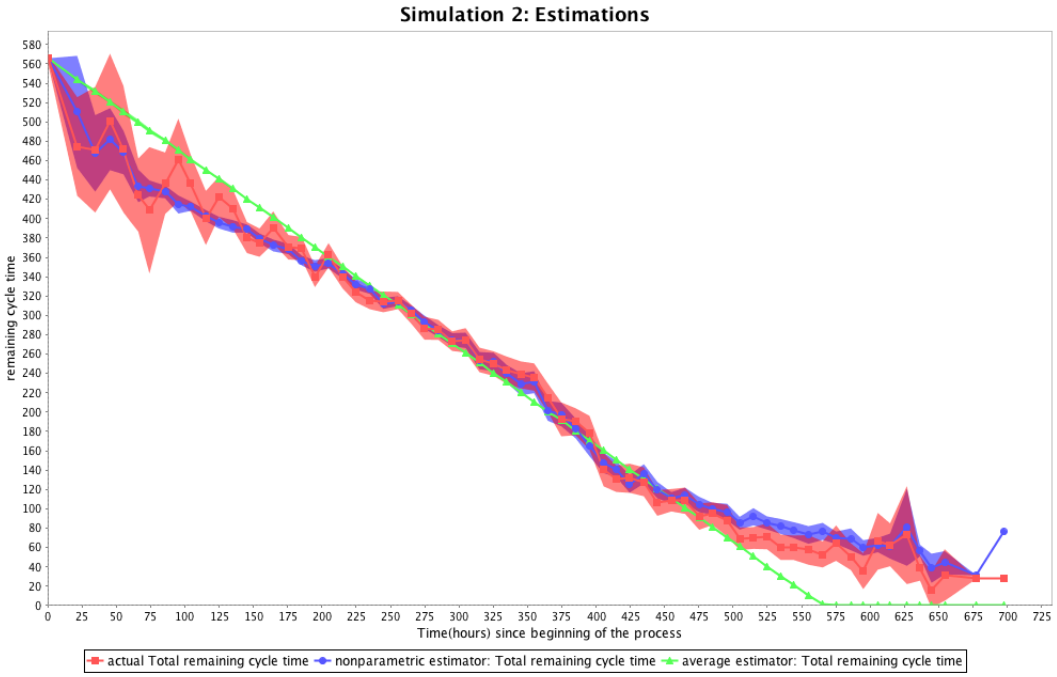


Figure 28: Total remaining cycle time - Simulation 2: parallel process

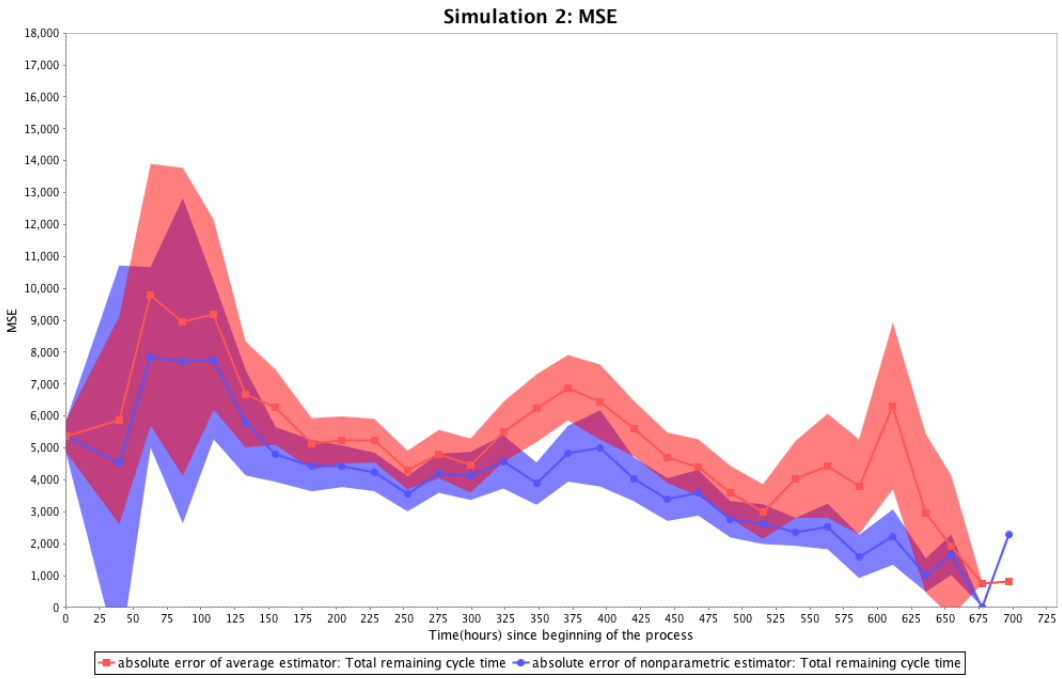


Figure 29: Mean Square Error - Simulation 2: parallel process

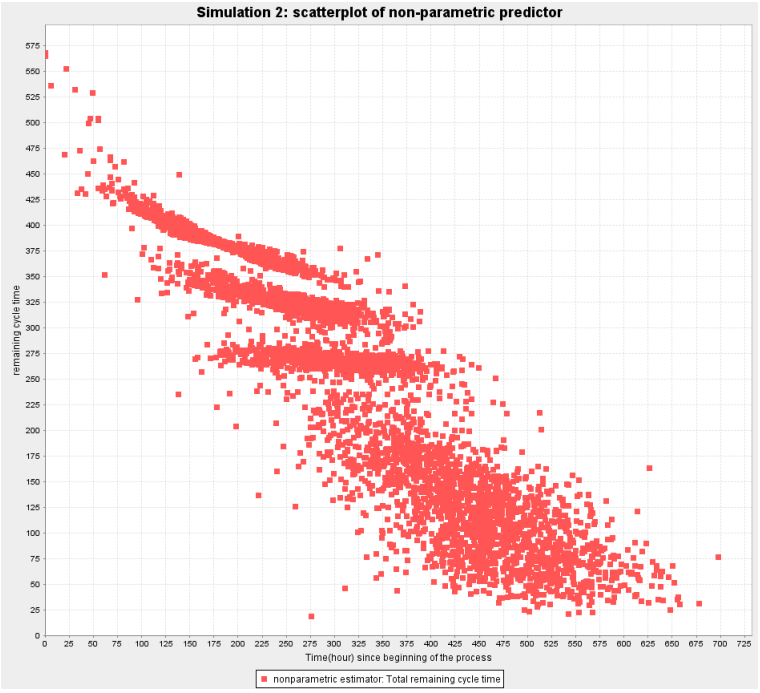


Figure 30: Simulation 2: scatterplot non-parametric predictor

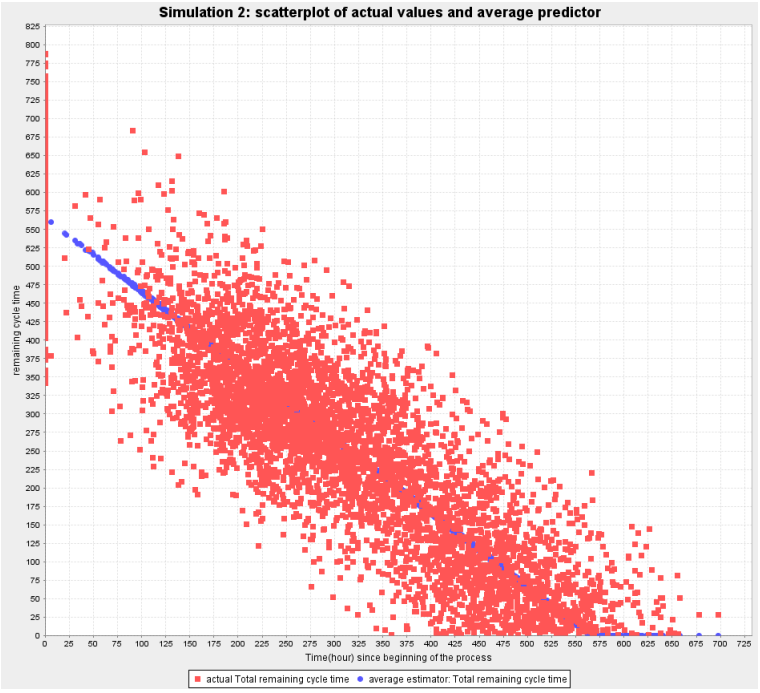


Figure 31: Simulation 2: scatterplot actual values and average predictor

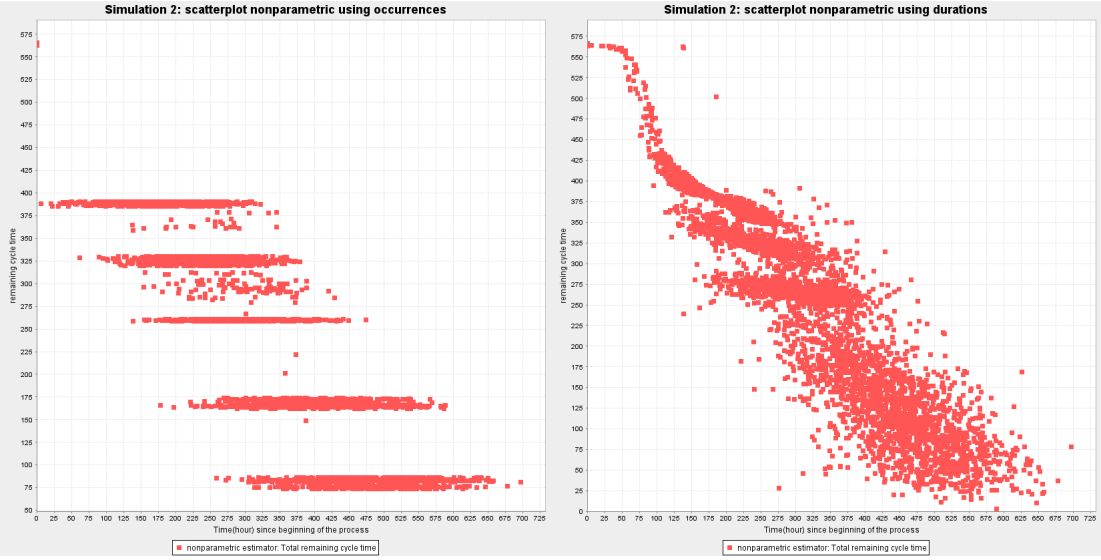


Figure 32: Simulation 2: scatterplot non-parametric predictor using occurrences (on the left) and durations (on the right).

by the linear shape of the scatterplot. There is an average cycle time of 24 days, through which the average predictor works. Note that, although each of the parallel sequences $A \cdot B1 \cdot C1$, $A \cdot B2 \cdot C2$ and $A \cdot B3 \cdot C3$ is on average 20 minutes long, the total average cycle time is slightly longer, 4 minutes long to be precise. Because the process is parallel, the total cycle time is the maximum of three different normal distributions[4], as a result the probability of the total cycle time being larger than 20 is greater than the alternative.

Figure 30 shows the plot of the non-parametric predictor, by comparing the estimations of the average predictor to the estimations of the non-parametric predictor, we get an insight into how the non-parametric predictor has a lower mean square error. Remember that the non-parametric predictor only compares cases and returns the local average of the remaining cycle time of the most similar cases. If the first activity of a case takes a long time, for example 14 days, the average predictor will predict a very straightforward 10 days remaining. However, the non-parametric predictor will construct a local average, based on all cases with the same sequence and the most similar durations for each activity. The prediction is therefore based on what activities they have finished and what time each activity took, which is a more realistic bases for a prediction.

In Figure 32 the scatterplots of the non-parametric predictor using occurrences (on the left) and the non-parametric predictor using durations (on the right) are shown. This figure shows that the non-parametric predictor using occurrences predicts the same remaining cycle time for each prefix-length, meaning that the prediction is not based on the time

[4] $E(\max(x, y)) > \max(E(x), E(y))$

spent but on the location in the process. The non-parametric predictor using durations shows clusters that are a bit similar. This is because any duration longer than zero indicates that an activity has occurred, meaning the non-parametric predictor using occurrences can also distinguish occurrences. However, the shape of the scatterplot of the non-parametric predictor using durations is more like a combination of the the average predictor and the non-parametric predictor using occurrences. The shape suggests that the remaining cycle time is dependent on the time spent. This is best explained by an example. Imagine that the activities $B1$ and $B2$ take a very short time to finish, because the total cycle time of the complete process is the maximum of three normal distributions, the probability of the rest of the case taking much more time than the time spent so far, is also greater. This shows that because the non-parametric predictor returns a weighted average, the complex behavior of the process contained within the logged data is not mimicked, but used to predict as accurately as possible.

Discussion Simulations 1 and 2 with respect to Hypothesis 1 These simulations show that there is always a set of predictor variables for which the non-parametric predictor performs similar or better than the average predictor, in fact the simulations show that the non-parametric predictor with the best set of predictor variables is at least as accurate as the average predictor and in many cases more accurate. All four simulations are considered to be worst-case-scenario's for the non-parametric predictor. Therefore, if the non-parametric predictor with the best set of predictor variables performs similar or better in these simulations, we can assume that there is a set of predictor variables for which the non-parametric predictor with the best set of predictor variables performs similar or better than the average predictor in all processes.

5.2.2 The effect of choice in a proces

If a process has any form of choice, there is more than one path that can be taken in this process. As different paths result in different cycle times, the variation in the total average cycle time increases. And as the variation increases, the accuracy of the average predictor decreases. In the worst case scenario the choice is a *deferred choice*, meaning there is no variable indicating which path will be taken. The alternative is *exclusive choice* in which the choice is known at forehand, or can be derived from the available data. Although the non-parametric predictor can not distinguish which path will be chosen, it can distinguish which path has been chosen, resulting in different predictions for different paths. Therefore, the non-parametric predictor is a better predictor in processes with choices than the average predictor.

Hypothesis 2. *Choices in processes that result in different cycle times increase the accuracy of the non-parametric predictor relative to the accuracy of the average predictor.*

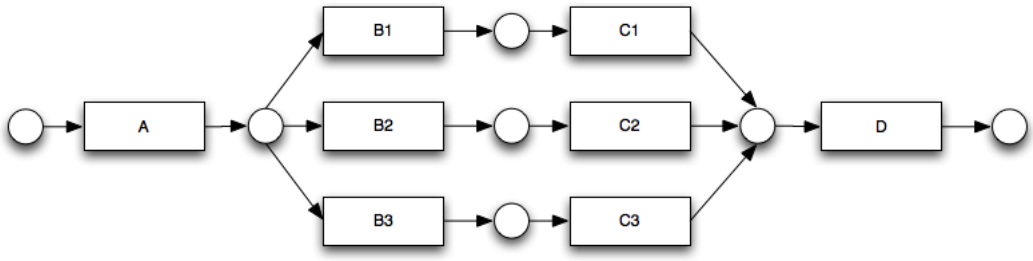


Figure 33: Simulated Process 3

Simulation 3 Figure 33 shows a process with choice, the probability of the choice is $\frac{1}{3}$ that either choice is taken. All activities have independent normally distributed process times, and sequences $B1 \cdot C1$, $B2 \cdot C2$ and $B3 \cdot C3$ have different flow times. To be precise, the activities $B1$ and $C1$ are on average 3 days with a standard deviation of 1 day, $B2$ and $C2$ are on average 10 days with a standard deviation of 6, and $B3$ and $C3$ both have an average of 20 days and 6 days standard deviation. Furthermore, activity A has a flow time of 20 days and 6 days standard deviation and D has no flow time. This process is simulated both with and without predictive data-attributes, i.e. the process is simulated for deferred choice, and exclusive choice. Figure 54 in Appendix B shows the CPN-Tools model used to simulate this process.

By applying both the non-parametric and the average predictor to the log of this process, and comparing the results, the hypothesis is tested. Note that after activity A is completed, before the choice is made, both predictors should have similar accuracy as the average is the best guess at this point.

Table 16: Results Simulation 3 : process with choice

Predictor	**Variables used**	**Mean Square Error** ± 95% confidence interval	
Average Predictor	-	144706.06	± 6579.22
Non-parametric Predictor	only activity durations	101298.57	± 5420.18
Non-parametric Predictor	only activity occurrences	78143.24	± 5182.85
Non-parametric Predictor	occurrences+durations	77524.78	± 5136.86
Non-parametric Predictor	all	20295.47	±1673.97

Results Simulation 3 The results in Table 16 shows that the non-parametric predictor is more accurate than the average predictor. In Simulation 1 and Simulation 2, the difference in the accuracy of average predictor and the non-parametric predictor was in the

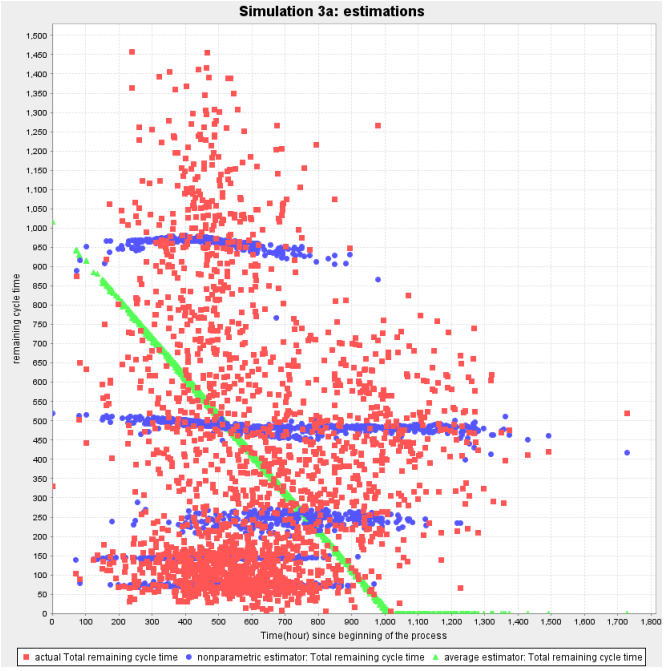


Figure 34: Scatterplot Simulation 3a with non-parametric predictor using attributes, occurrences and durations, in which the choice is known beforehand(exclusive choice)

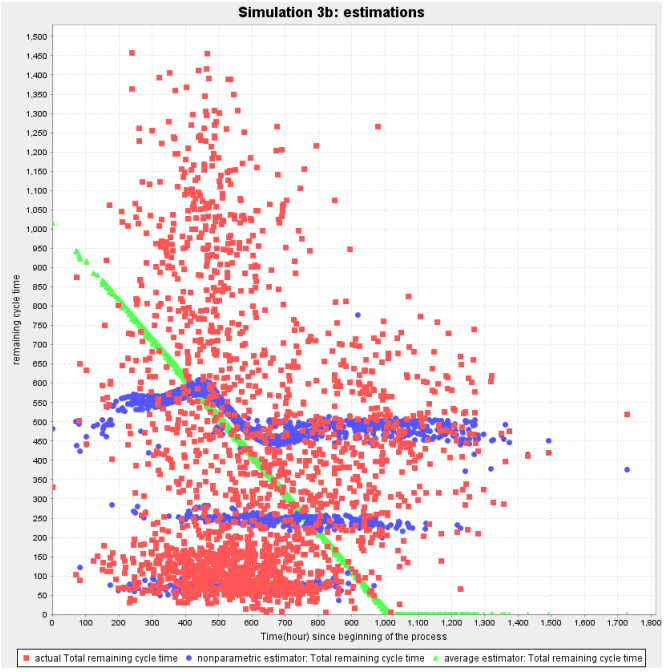


Figure 35: Scatterplot Simulation 3b with non-parametric predictor using occurrences and durations, in which the choice is not known beforehand(deferred choice)

majority of results not significant. The relative difference in the accuracy of the predictors in this simulation compared to the relative difference in accuracy of the predictors in simulations 1 and 2 is very large. In this simulation the best performing non-parametric predictor performs about 6.5 times more accurate than the average predictor, in simulations 1 and 2 the non-parametric predictor was between 1.2 and 1.4 times more accurate. Therefore we can confirm the hypothesis that choices in process that result in different cycle times increase the accuracy of the non-parametric predictor relative to the accuracy of the average predictor. Note that Simulation 3a and Simulation 3b are the same simulations, using different predictor variables. Simulation 3a is result of a non-parametric predictor with a data-attribute indicating the choice, and Simulation 3b is the result of a non-parametric predictor without that data-attribute.

Discussion Simulation 3 Because both Hypothesis 1 and Hypothesis 2 have been confirmed, we know that the non-parametric predictor performs at least as good as the average predictor, furthermore the non-parametric predictor performs much better than the average predictor as the variance in the cycle times increase as a result of choice in the process.

Figure 34 shows a scatterplot of the actual values, the predictions of the average predictor as well as the non-parametric predictor (based on all data) for exclusive choice, and Figure 35 shows a scatterplot of the actual values, the average predictor and the non-parametric predictor (based on only occurrences) for deferred choice. Note that only the values of the non-parametric predictor are influenced by the presence of information on the choice.

Figure 35 shows the scatterplot of the non-parametric predictor for the deferred choice, in this scatterplot the non-parametric predictions can be divided in three different (horizontal) clusters are visible, and a line that follows the average predictor. The non-parametric predictor in this simulation uses the occurrences to determine what choice was made. Therefore it shows three distinct clusters of predictions, one for the sequence $A \cdot B1$ around 480 hours, one for $A \cdot B2$ around 240 hours and one for $A \cdot B3$ around 72 hours. More interestingly is the line of non-parametric predictions that follows the average predictor. This line of dots contains the non-parametric predictions after activity A, at this point in the process the occurrences have no information. The only information at this point that differs is the duration of A. Remember that the average predictor predicts using the total average cycle time minus the time spent so far, resulting in a straight line. Because the non-parametric predictor has only the duration of activity A to compare cases, the prediction becomes dependent on the time spent, which results in predictions that resemble the average predictor.

Figure 34 shows the scatterplot of the non-parametric predictor for excusive choice. In this scatterplot there are 6 clusters, of which two overlap. Three of these are the same clusters as in Figure 34, and the other three are the predictions after activity A. Because there is a data-attribute present that describes which choice is made, the non-parametric predictor can make three different predictions, one around 960 hours, one around 480 hours, and

one around 144 hours. Meaning that around 480 hours there are two clusters, which is the reason there are as many dots around 480 hours. This shows that the non-parametric predict uses the data-attribute in combination with the occurrences to create different predictions for different types of cases.

This simulation perfectly illustrates how the non-parametric predictors compares cases based on the predictor variables. The difference in exclusive choice and deferred choice shows how the non-parametric predictor constructs a local average. In the exclusive choice, the non-parametric predictor distinguishes three different local averages from the start. In the deferred choice, the non-parametric predictor initially can not distinguish the different choices, only after the choice is made is the non-parametric predictor able to distinguish the three different types of cases.

5.2.3 The effect of loops in a proces

A loop, like a choice, can be predictable or unpredictable. In a process with a loop that is data-based, the number of loops is dependent on some predetermined data. Assuming this data (on which the number of loops to come is based) is available for the non-parametric predictor, it can distinguish cases based upon this data. Therefore the predictor makes different predictions for cases in which we expect a different number of loops.

However, in a process for which the loop is based on a random choice, we can not distinguish cases in advance. The non-parametric predictor can only distinguish how many times the process has looped so far. With this information the non-parametric predictor constructs a local average over all measurements with at least as many loops, while the average predictor is dependent on the total average number of loops. For example, a partial case with 3 loops so far is compared to all *measurements* with 3 loops. Since all measurement with 3 loops correspond to all *cases* with at least 3 loops, the non-parametric prediction is based on a better subset than average prediction. This leads to the following hypothesis.

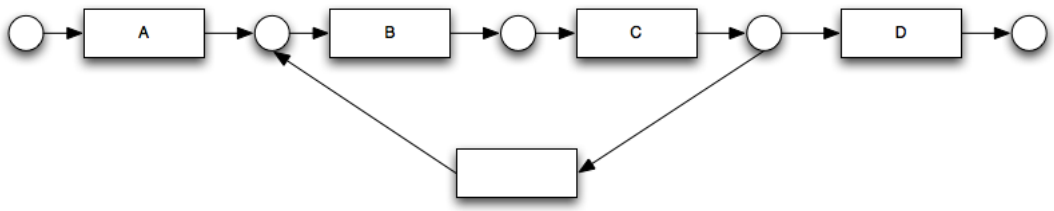


Figure 36: Simulated Process 4

Hypothesis 3. *Loops in processes that result in different cycle times increase the accuracy of the non-parametric predictor relative to the accuracy of the average predictor.*

Simulation 4 Figure 36 shows a process with a loop, in this process all activities have independent normally distributed process times that are not significantly different. There are two different simulations for this process.

Simulation 4a: In this simulation, activity A determines the number of iterations and stores this value in an data-attribute. As a result, the log contains the number of occurrences of each activity, the average duration of each activity, and the total number of iterations. The number of iterations for this process has a maximum of 11, and each possible number of iterations $(1 \ldots 11)$ is selected with an equal probability of $\frac{1}{11}$.

Simulation 4b: In this simulation, the loop is based on choice with a probability of $\frac{1}{2}$. This process has therefore no theoretical maximum number of iterations. Note that there is no information available in the form of a data-attribute on which the choice will be made.

Table 17: Results Simulation 4a : process with a loop, exclusive choice

Predictor	**Variables used**	**Mean Square Error** ± 95% confidence interval	
Average Predictor	-	7069.55	± 105.18
Non-parametric Predictor	only activity durations	7821.55	± 148.42
Non-parametric Predictor	only activity occurrences	4924.50	± 83.74
Non-parametric Predictor	only attributes	5095.58	± 87.25
Non-parametric Predictor	all	45.05	± 1.21

Table 18: Results Simulation 4b : process with loop, deferred choice

Predictor	**Variables used**	**Mean Square Error** ± 95% confidence interval	
Average Predictor	-	2784.02	± 163.69
Non-parametric Predictor	only activity durations	2647.49	± 151.10
Non-parametric Predictor	only activity occurrences	1893.30	±121.51
Non-parametric Predictor	both	2276.43	± 130.43

Results Simulation 4 The results in tables 17 and 18 show that the non-parametric predictor is significantly more accurate than the average predictor in both simulations. Figures 37 and 39 show scatterplots of simulations 4a and 4b, Figures 38 and 40 show scatterplots of the actual remaining cycle times and average predictions of both simulations.

Discussion Simulation 4 In Simulation 4a, the accuracy of non-parametric predictor using all predictor variables, compared to the other predictors, is more than a hundred

times higher. The explanation of this huge difference lies not in the high accuracy of the non-parametric predictor using all variables, but in the low accuracy of the other predictors. Because the process contains a loop, the variation in cycle times is large, which results in a poor performing average predictor. The non-parametric predictor using durations is based upon the average duration of activities, which is even less useful than time spent so far. The non-parametric predictor using occurrences can count the occurrences and compute better average estimations, however it has no information on the remaining number of iterations. On the other hand, the non-parametric predictor using attributes has information on the total number of loops it will have, but no information on the current state. By combining the attributes and the occurrences, the non-parametric predictor has information on both the current state and the total number of iterations, as a result it can accurately predict the remaining cycle time for a case.

In Simulation 4a, there are a predefined number of iterations (minimum 1, maximum 11), where each number iterations will occur with equal probability. This means that there are 11 different types of cases, each case has a predefined number of iterations. Figure 38 shows the actual values and the average predictions in a scatterplot, in this plot the 11 different types of cases are visible in the form of lines of clusters. Because the total number of iterations for a case, and the occurrences of activities is known, the non-parametric predictor can distinguish each type of case. Figure 37 shows the scatterplot of the non-parametric predictor, in this plot it is visible how the non-parametric predictor constructs a local average for each prediction. As a result, the error of the non-parametric predictor is directly related to the variation in the cycle times. Whereas the average predictor constructs a single average prediction for all cases, which means that the average predictor will predict that each case will iterate 5 times.

The difference between Simulation 4a and 4b, is that in Simulation 4a each number of iterations has an equal probability of occurring. In Simulation 4b however, 0 or 1 iterations are the two most likely possibilities, as the iteration is based on deferred choice. This has two results that decrease accuracy of the non-parametric predictor; an unpredictable number of iterations, and outliers. Because there is no information on the number of iterations that each case will have, it is impossible to predict the remaining cycle time with an accuracy as high as that of the non-parametric predictor in Simulation 4a. The outliers that are created as a result of the structure of this process introduce even more variation in the remaining cycle times, thereby increasing the error.

Table 18 shows that the non-parametric predictor is better than the average predictor. Although the non-parametric predictor can not predict the number of iterations, it can distinguish that the amount of iterations so far. Figure 39 gives us an insight into the behavior of the non-parametric predictor in this simulation. Basically it will return a prediction based on the location in the process, also it will predict about the same value at the same point in the process. Note that it will not predict exactly the same value, as the local average suffers from the increase of variation with each iteration. This behavior is visible in Figure 39 by the clusters of predictions that all have the same prediction but a

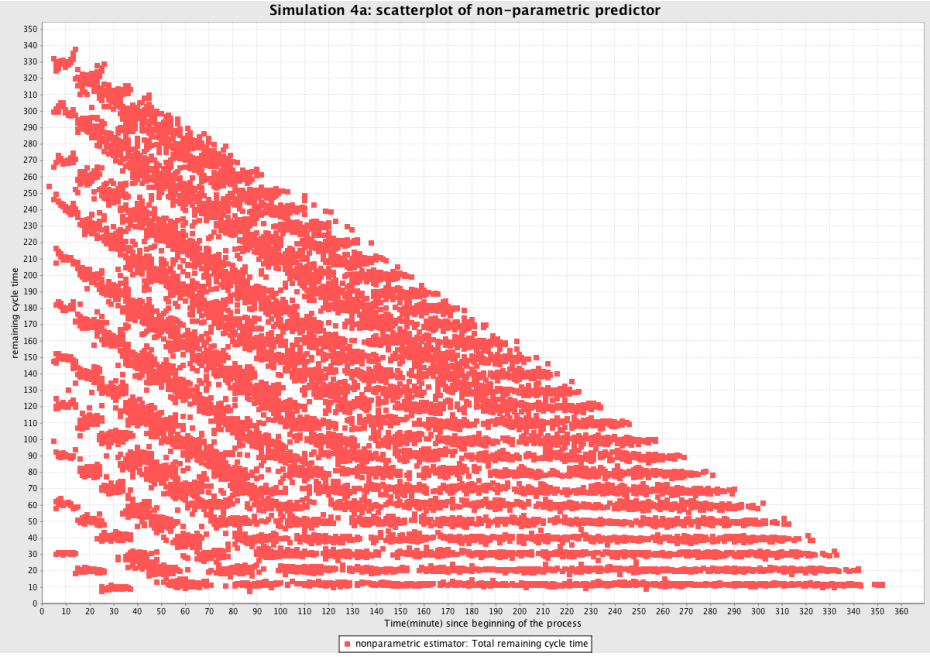


Figure 37: Scatterplot Simulation 4a: non-parametric predictor

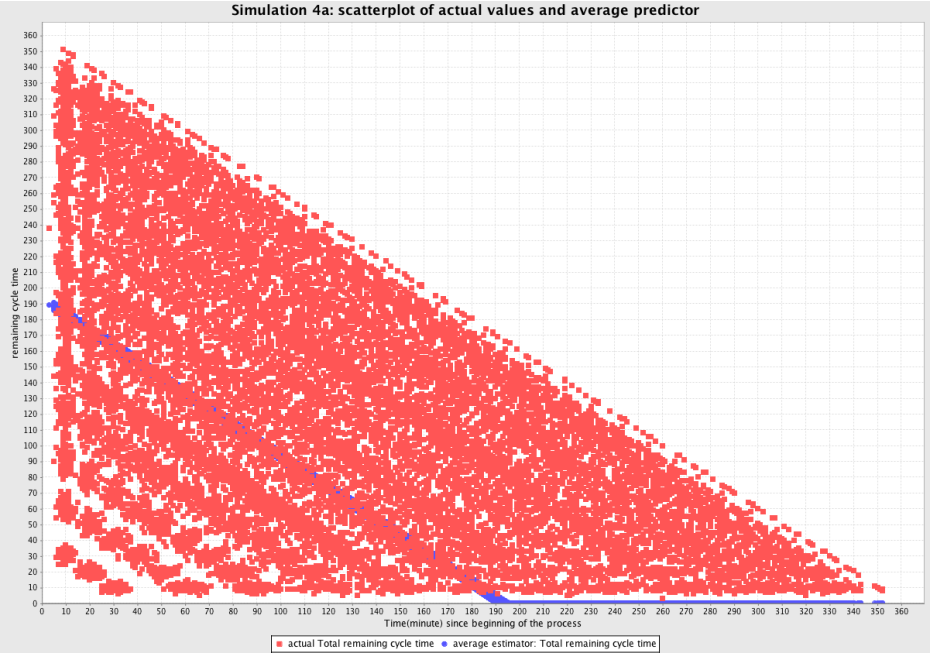


Figure 38: Scatterplot Simulation 4a: actual values and average predictor

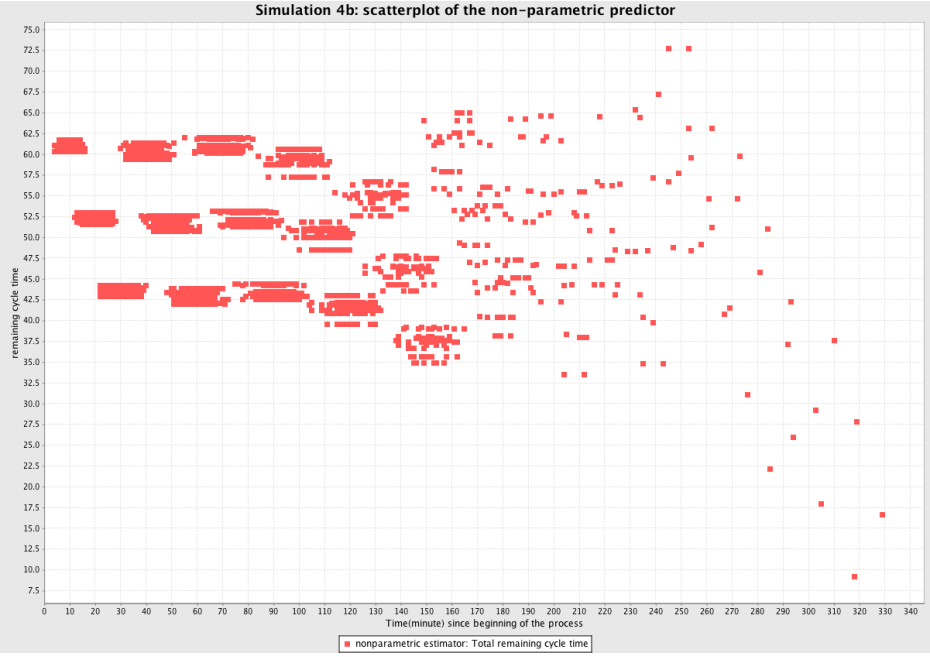


Figure 39: Scatterplot Simulation 4b: non-parametric predictor

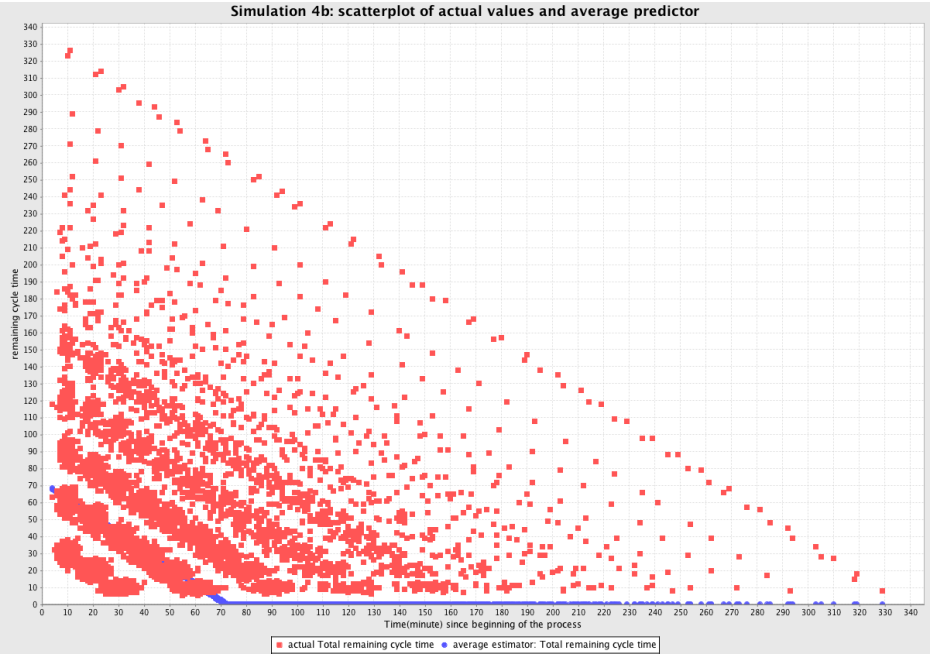


Figure 40: Scatterplot Simulation 4b: actual values and average predictor

different time spent. The shape and size of the clusters is and indiction of variation, which increases after each iteration.

These simulations show that the non-parametric predictor compares cases on a combination of all predictor variables. If the non-parametric predictor compares a case on only the occurrences or only the data-attribute that indicates the number of loops, it would never have the accuracy shown in Simulation 4a. It is important to understand that the non-parametric predictor does not use the value of the data-attribute to derive the number of iterations, rather it categorizes all cases based on the value of that data-attribute. Also it categorizes cases based on the occurrences of activities. The combination of those two pieces of information results in local averages that accurately predict the remaining cycle time.

5.3 Summary of the case study and the simulations

The case study suggests that the non-parametric predictor performs better than the average predictor in case taken from practice. Furthermore the case study shows that the non-parametric predictor is capable of selecting relevant predictor variables and ignoring irrelevant predictor variables, however, the non-parametric predictor does suffer in accuracy from the presence of irrelevant predictor variables.

The simulations suggest that the non-parametric predictor is at least as good as the average predictor. This was illustrated using two processes that are considered to be the worst case scenario for the non-parametric predictor. The simulations also show that the performance of the non-parametric predictor is relatively better than the performance of the average predictor if loops or choices are present, especially if these loops or choices are predictable by e.g. a data-attribute. More importantly, the simulations illustrate the working of the non-parametric predictor, especially how the local average works.

From both the case study and the simulations, we conclude that the non-parametric predictor is a better predictor than the average predictor. However, the accuracy of the non-parametric predictor can be increased by intelligently selecting a subset of predictor variables, and perhaps designing the process such that there is information on choices included in the data-attributes.

6 The software

6.1 Introduction

The work presented in this thesis was implemented in the process mining framework ProM [14]. ProM is a process mining framework that offers various process mining methods and can easily be extended with plugins. The ProM framework, combined with the ProM-import application, provides facilities for easy import, export, filtering or checking of logs.

Two plugins were developed; the *prediction plugin* and the *visualization plugin*. The prediction plugin can use a log to test predictions by the use of cross-validation after which it writes the predictions back to the log. It is also able to simply process a log and writing the optimum bandwidth back to that log. The plugin also provides an interface for single predictions for other plugins. After the predictions are tested and written back to the log, the visualization plugin uses the log in which the predictions are written to visualize the predictions in charts. The log with only the bandwidth in it, is used by the *Recommendation service* [30], this is a plugin that provides a server for other workflow engines to connect to and ask for recommendations concerning a single case. The Recommendation service also provides predictions to other workflow engines by asking for a single prediction of the prediction-plugin. Based on the log with the bandwidth in it, and a single case, the prediction-plugin returns a single prediction.

6.2 Programming environment

As sir Isaac Newton wrote *"If I have seen further, it is by standing on the shoulders of giants."*, it is in this spirit that both the work presented in this thesis and the software are based on the work of others. More specifically on ProM, the R-project, the NP-package for R, and JFreeChart.

ProM The ProM framework was developed by the Process Mining group[5] at the University of Technologies Eindhoven, a detailed description is given in [14]. Process mining is a technique with which one can derive information from an event-log on the process that created the log. It is possible for example to retrieve the workflow model of the information system as it is in reality, based on the sequence of events in the log. ProM is a framework that enables a user to import a log and analyze it in various ways with the help of plugins. Through the use of plugins the ProM framework is continuously growing in functionality. The ProM framework, as well as its plugins, were developed in Java. This language enables Object-Oriented programming and is platform-independent. As a result ProM is highly extensible, and available for all platforms. As a framework, ProM enables a programmer to write

[5] http://www.processmining.org

a plugin without worrying about importing the log, exporting the log, etc. As the prediction method described in this thesis is also based on a log, the ProM framework provides an excellent basis for it.

A part of ProM that we specifically rely on is the Recommendation service as described in [30]. The Recommendation service is designed to connect to other workflow software (specifically the workflow engine 'Declare') and give recommendations on choices that need to be made in currently active cases. This is relatively similar to the prediction engine that returns e.g. remaining cycle times on currently active cases. Both the recommendation service and the prediction engine require a log and a new (partial) case as input, and return some piece of information. The recommendation service is flexible enough to use the prediction-plugin in order to provide a single prediction to the workflow system it is connected to. We have therefore chosen to use the recommendation service as a means to connect to workflows to provide predictions.

R with NP The non-parametric regression (see Chapter 3) used in the prediction method, has already been implemented as a package for R called NP [19]. R [6] is an open source statistical language and environment developed to be similar to S [8], a statistical language and environment developed by Bell Laboratories for statistical research. R provides two main techniques to connect to Java. The first is called JRI and uses R as a dynamic library, the second is Rserve, of which there is a server and a client version enabling Java to connect to R on a server via TCP/IP. Although a wrapper is present to enable a connection via both, the server/client implementation is the preferred choice to enable use of a grid-machine. The NP package, is one of the many statistical packages available for R. The non-parametric regression as implemented by NP, is an exact match to the regression techniques as described in [25]. This is also the main reason that R with NP was chosen as a basis for this plugin.

JFreeChart JFreeChart[7] is a package for Java that enables the creation of various forms of charts, to visualize the accuracy of the predictions some charts are created with the help of this package.

[6] http://www.r-project.org
[7] http://www.jfree.org/jfreechart/

6.3 The prediction plugin

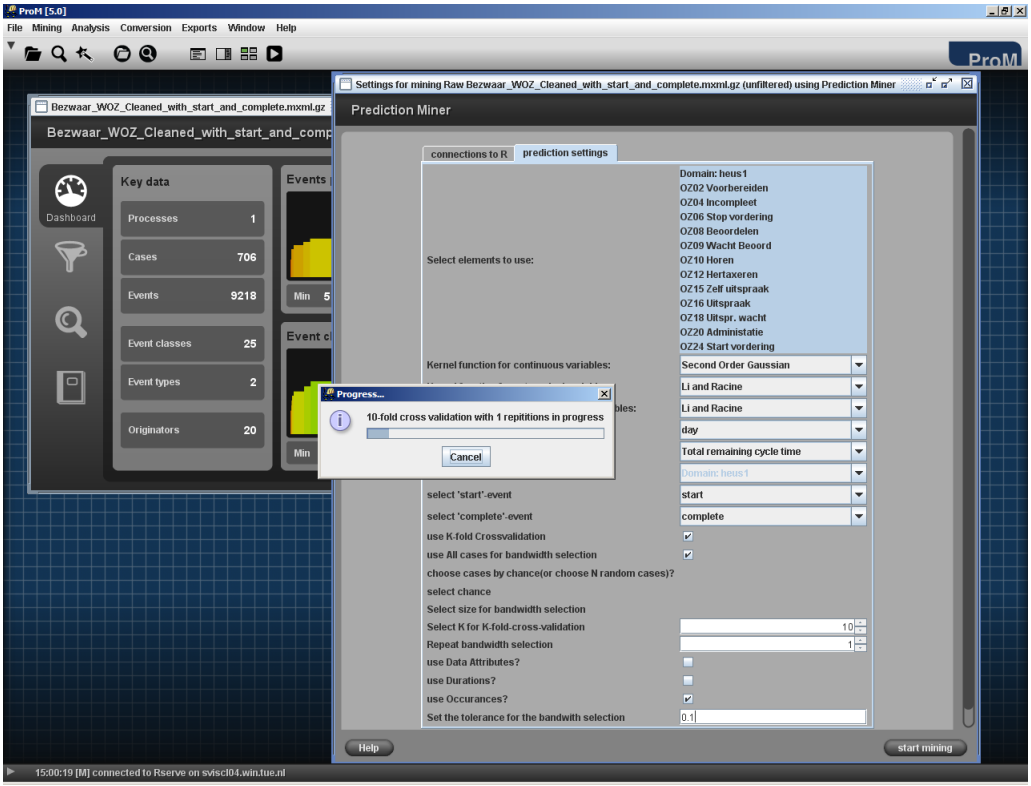


Figure 41: Screenshot of ProM with prediction plugin running

The prediction plugin has 2 graphical parts, the settings window and the results windows which is basically a windows with a lot of tables. Figure 41 shows a screenshot of ProM with the settings window of the prediction plugin and a progress bar, the progress bar indicates that the prediction plugin is already running. ProM is responsible for the the main user-interface, loading the log, loading the plugin, etc. The settings windows is the main user-interface of the plugin, an overview of settings is shown in Table 19.

Although most settings are clearly related to aspects of the prediction method described in previous chapters, there are some settings that we have not yet discussed. The first settings “elements to use”, is an option to filter certain activities from the measurements. ProM has a filter-function built-in, however the filter ProM uses completely removes the activity from the log, which results in a different cycle time if the activity was at the begin or end of the case. The “type of time measurement” setting is used to choose the format of the predicted remaining cycle times. The settings ”start event” and ”complete event” select which events in the log determine the start and end of an activity. The log, of which an example is shown in Figure 6, consists of events. By default these events have

types like “start” and “complete”, however in some logs they are named differently. The start-event and complete-event of one specific activity are used to determine the duration of that activity.

The option ”choose cases by probability” selects the method of choosing the subset of cases for the bandwidth selection. As explained in Chapter 3.6, the bandwidth selection is of exponential complexity. Therefore a subset of the measurements is used to keep the time to compute the optimal bandwidth feasible. Two methods of choosing a subset for the bandwidth were implemented, one method selects S cases at random, the second method selects cases with a probability p, and thus probably selects $p \times N$ cases, N being the total number of cases.

The setting ”repeat bandwidth selection” is related to option to use a subset of the cases for bandwidth selection. As the cases are chosen randomly, it is possible that the subset of cases (randomly) chosen for the bandwidth selection is a very poor choice. By repeating the cross-validation, the probability of a poorly chosen bandwidth affecting the results is reduced. The last setting is the ”tolerance for the bandwidth selection”, this is a setting that is necessary for the NP package. The method we use to optimize the bandwith requires a value that determines when the optimization function has reached an optimum. The lower this value is set, the longer the optimization will continue. Therefore this is also a setting to reduce computation time.

The prediction plugin is relatively straightforward in use. The plugin has a log as input, a GUI then prompts for settings after which the log is processed and the measurements are extracted. Based on what settings the user has selected, the measurements are exported to R in order to apply the non-parametric regression by use of the NP package. If a K-fold cross validation was selected in the settings, the results are shown in a table for the user to inspect and the predictions (non-parametric and average) are written back to the log. For the purpose of analysis, the predictions of both the non-parametric and the average predictor, the actual values and the absolute and relative error are written into the log, which can in turn be shown in charts by the visualization plugin. Otherwise, if no cross-validation was required, the optimum bandwidth is calculated only once over the complete log, which is then written back to the log. The log with only the bandwidths can then be stored by the recommendation engine, which can call for a single prediction given a log with bandwidths and a single case.

The communication with R is done by either JRI or Rserve, the first is a rather straightforward method of calling the locally installed R software. The second method is a TCP/IP enabled method, of communicating with an R-server, which enables distributed calculation of the K-fold cross-validation. Each of the K regression analysis is an independent sequence of calculations, meaning they can be easily done in parallel. By running Rserve-server on multiple computers it is possible to connect to multiple servers on multiple computers thus running a K-fold cross validation a maximum of K-times faster.

Table 19: Overview of the Settings used by the prediction plugin

Setting	Options
elements to use	<activities in the process>
kernel function for continuous values	[Second Order Gaussian, Second Order Epachanikov, Uniform]
kernel function for categorical values	[Aitchison and Aitken, Li and Racine]
kernel function for ordered discrete values	[Wang and van Ryzin, Li and Racine]
type of time measurement	[second, minute, hour, day]
predict the	[total remaining cycle time, remaining cycle time until element <..>, occurrence of element <..>]
start event	[start]
complete event	[complete]
use k-fold cross-validation	[true,false]
use all cases for bandwidth selection	[true,false]
choose cases by probability	[true,false]
probability for the bandwidth selection	<number>
size for the bandwidth selection	<number>
k for k-fold cross-validation	<number>
repeat bandwidth selection	<number>
use data-attributes	[true,false]
use occurrences	[true,false]
use durations	[true,false]
tolerance for the bandwidth selection	number

6.3.1 Limitations and design choices

As every piece of software, there are some limitations to the implementation as opposed to the theory of predictions.

Constant variables The NP package provides us with many highly optimized non-parametric regression functions. However, the function used in our software does not allow one predictor variable to be constant. As the NP package is normally used for a single regression analysis, this feature is understandable. In those circumstances the regression is run only once, and the user can simply remove the constant variable. However, the prediction method we discuss is automated and requires the software to continue, even in the presence of constant predictor variables. In order to deal with this the prediction plugin removes all constant variables from the regression-analysis it sends to R and NP. When e.g. 10-fold cross validation is used on a log, each regression-analysis uses a different 10% of the log, it is therefore possible for a regression variable to be constant in only one regression analysis. This means it might be possible that each regression analysis in a cross-validation is on a different set of variables. In general this does not change the results much as the regression variables that are constant are usually completely smoothed out, meaning they have almost no influence. Because of this, the bandwidth variables are only exported to the log after a single regression analysis (no cross validation) which is used for the recommendation engine.

Bandwidth selection As shown in Section 3.6, the bandwidth selection procedure is of exponential complexity. This basically means that calculating the bandwidth gets exponentially slower in the amount of variables, the size of the log, and the domainsize of the bandwidth. In order to keep calculations within reasonable time, a setting is added to the program to choose a subset of size n of the cases on which to calculate the bandwidth. This 'bandwidth-set' is randomly chosen either by probability p (e.g. 0.2 leading to probably $N \cdot 0.2$ cases) or by randomly choosing n cases. These cases are chosen before the measurements are extracted from them, otherwise it is possible to get a bandwidth-set with for example only empty cases.

Rounded occurrence predictions In Chapter 4 we described an occurrence predictor that rounds the results of the regression functions. Altough we were able to round the predictions themselves, we were unable to alter the bandwidth selection to internally round the predictions. Therefore the bandwidth is selected to be optimal for a continuous prediction and not for the rounded predictions. This might result in some loss of accuracy.

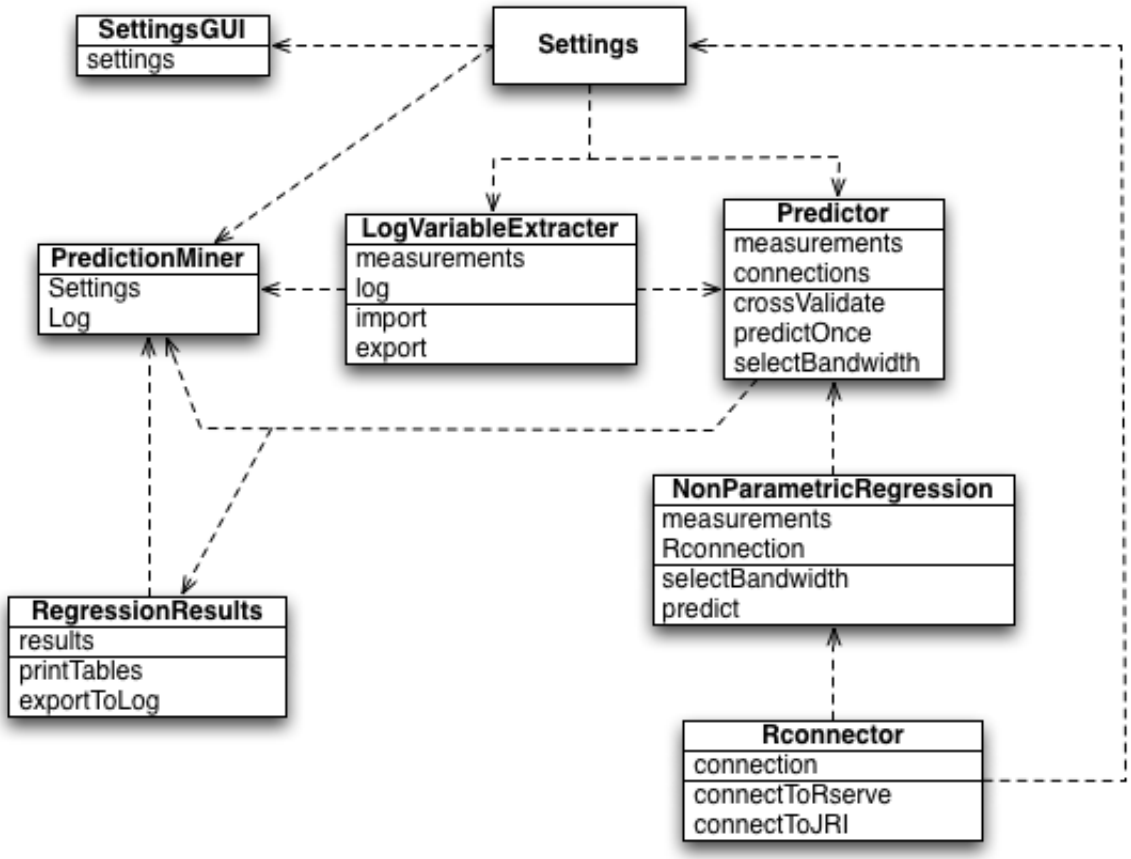


Figure 42: UML diagram of the prediction plugin

6.3.2 Architecture

The architecture of the prediction plugin is straightforward, as shown in Figure 42. Note that some minor classes were left out this diagram to prevent clutter. The *Settings*-class is a static class that can be accessed by all other classes when needed, this class is initialized by the settingsGUI. The predictionMiner is the main class that is called after the settingsGUI by ProM, in Figure 43 the sequence in which this class calls all other classes is shown.

Each of the classes shown has its own reason of existence, the function of each class is clear from the name. The *LogVariableExtracter* for example derives the measurements from the log. As the only function with access to the log, it is also responsible for writing back variables to the log. A more interesting relation is that between the *Predictor*-class, the *NonParametricRegression*-class and the *Rconnector*-class. The *Predictor*-class makes sure that measurements are cross-validated, or not. It combines the right variables for the regression functions, so that the regression part in the *NonParametricRegression*-class can

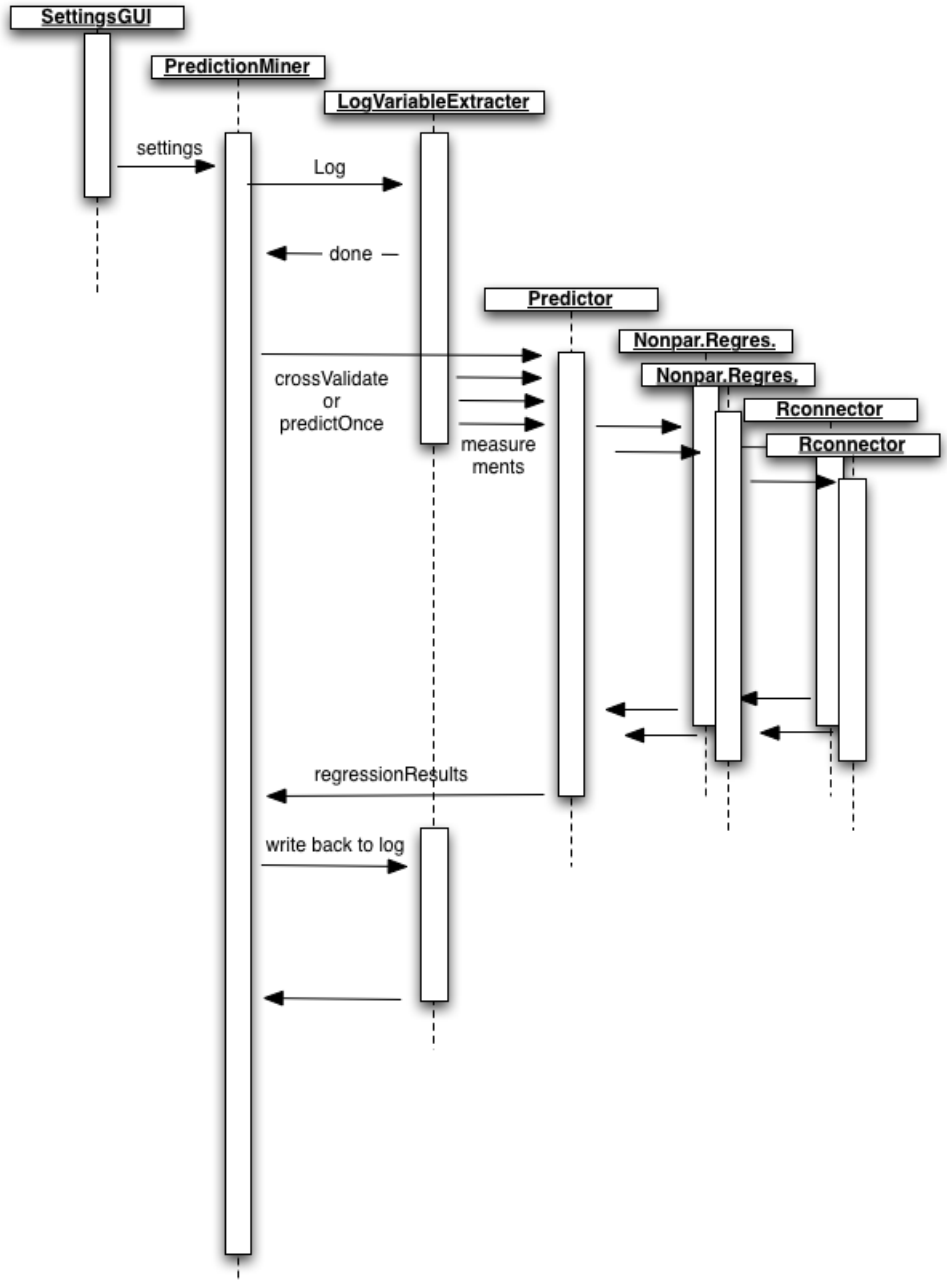


Figure 43: message sequence diagram of the prediction plugin

be run as separate threads. As mentioned before, the software can connect to a server with Rserve by TCP/IP. This enables us to run a K-fold cross-validation on (maximum K) multiple servers in parallel.

7 Future work

During the development of this prediction method, the problems encountered give rise to various solutions. However, for practical reasons, the most optimal solution cannot always be chosen. In order to encourage others to continue researching the use of non-parametric regression as a prediction method for cycle times, we give some suggestions for future work. Therefore we present some suggestions for future work here, with a short description of its usefulness.

Set-based comparison of categorical predictor variables In the current form of regression, a categorical predictor variable has only one value. However, in a process with loops a certain task might be encountered several times in a log. As explained in Chapter 3, the duration of a task like this are averaged and the occurrences are counted as one would expect. A categorical predictor variable like the name of the person (e.g. "Mike") who was responsible for the task can also change, overwriting the name of the person (e.g. "Tim") responsible for the previous iteration of this task. However, if categorical predictor variables could also be a set of values (e.g. {"Mike","Tim"}), a more realistic prediction model is obtained. In the current solution information is lost, and most likely some accuracy with it.

Automated redundant variable removal The most important part when creating a prediction model with non-parametric regression is the bandwidth selection process. A method to smoothed out (i.e. ignore) irrelevant predictor variables by allowing the bandwidth of the specific variable to approach 0, is shown in [25]. Based on this, [22] shows irrelevant predictor variables can automatically be removed, based on the bandwidth. Removing the irrelevant predictor variables would decrease computation time and increase accuracy. Although one could manually remove irrelevant predictor variables, an automated removal could be more accurate. The selection of variables could be done based on the relative size of the bandwidth, but also more straightforward statistical methods can be applied to enhance the prediction. Whatever method used, choosing a more relevant (sub)set of predictor variables will increase accuracy and decrease computation time.

Idle-time bandwidth optimization As the computers providing the predictions are usually not used during the nights and weekends, they can continue to optimize the bandwidth-variables. As over time information systems change, so do the models that describe them. By constantly optimizing the bandwidth, any trends or changes relevant to the prediction of the remaining cycle time will also be included in the non-parametric predictor.

Selection of the training-cases Over-time the number of recorded cases will grow, meaning that the bandwidth-selection will also become more complex. Also, a lot of cases might become obsolete over time, meaning they have no relation to the (evolved)

process anymore. Rules and regulations might drastically chance the process behind the information system. If a large set of cases that is not representative anymore to the current process are kept, they could seriously decrease the accuracy of predictions and harper performance. This issue could be resolved by selecting a set of cases on which to base the predictions that accurately describe all possibilities. Although this will be a big set of cases, an optimum can be found such that computations remain in a feasible order of time, while maintaining a high accuracy in predictions. Statistical methods (e.g. cluster analysis) can be applied to decrease the size of the training-cases without decreasing, maybe even increasing, the accuracy of the predictions.

Attribute prediction Like the prediction of occurrences and remaining cycle time, it also possible to predict the value of an attribute. This requires the attribute to be present for each case at each point in the process.

8 Conclusion

In this thesis we described a prediction method in the context of process mining, meaning it derives information from an event-log. The event-log is constructed based on the data stored by the information system. From the event-log we derive measurements, which are used as a bases for the non-parametric regression. Non-parametric regression is a form of regression that requires no assumptions on probability distributions, or other information on the data. The non-parametric regression model must be constructed once, after it has been constructed predictions can be made for any (new) case. Non-parametric regression works by constructing a local (weighted) average over the recorded remaining cycle time of the old cases that are most similar to the new case for which we need a prediction. A case study and several simulations show that the non-parametric prediction method performs similar to or better than the average predictor depending on the cycle time distribution and the structure of a process. The ProM plugin that implements the prediction method is used for both the case study and the simulations, this plugin is described in Chapter 6. In the previous chapter we presented some suggestions for future work, and in this chapter we discuss the results and their relevance.

Both the case study and the simulation in Chapter 5 show that the non-parametric predictor performs better than the average based predictor. The prediction method requires no deep knowledge of the process in order to use it, but if more knowledge on the process is available it can be used to optimize the data that the prediction method uses. Furthermore, no assumptions on the process are made, meaning that this prediction method can be applied to any type of process. For these reasons this prediction method is excellent for large complex processes. Because it is possible to simply start the process of optimizing the regression model without knowing anything of the process, wait until it finishes, and start using the prediction method. In short, because the prediction method requires very little knowledge of the process it is easy to just start using it.

However, despite these strengths, the prediction method has some limitations. Building the regression model, although required only once, is a very expensive computation in terms of time, more specifically the (algorithmic) complexity of the prediction method is exponential. The computation of the results shown in Chapter 5 took a long time, although the examples were small and the calculations were not as precise as desired. If the prediction method is used in practice, it will need to work on larger data-sets and compute a regression model with more precision. Note that there is the possibility of over-optimizing the model, although this is not investigated in this thesis. If the data-set is analyzed, cleaned and optimized at forehand, it would save a lot of time building the regression model. Also, because the regression model is built only once, it is possible to optimize the regression model while the information system is idle, which does not solve the problem but handles it in a more practical way.

In this thesis we have compared the prediction model only to the average based predictor,

however, there are many (more intelligent) predictors available. Before the prediction method is applied to any actual information systems, a comparison of the accuracy of different methods should be done. Every process is different, and every prediction method has a different approach with different strengths and weaknesses. It is very likely that for a particular type of process there is a "custom" made prediction method that performs much better than the non-parametric prediction method. However, many information systems exist that support extremely complex processes, or processes that are simply not important enough to justify a dedicated approach. Basically, the benefits of analyzing a process, with no certainty that it would actually result in a better prediction method, do not always outweigh costs. The non-parametric prediction method is in terms of costs a very cheap method, while in terms of computation a very expensive method.

References

[1] W.M.P. van der Aalst, B.F. van Dongen, C.W. Günther, R.S. Mans, A.K. Alves de Medeiros, A. Rozinat, V. Rubin, M. Song, and H.M.W. Verbeek. ProM 4.0: Comprehensive Supports for Real Process Analysis. In *Application and Theory of Petri Nets 2007*, volume 4546, 2007.

[2] 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 and Knowledge Engineering*, 47(2):237–267, 2003.

[3] W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.

[4] J. Aitchison and CGG Aitken. Multivariate binary discrimination by the kernel method. *Biometrika*, 63(3):413–420, 1976.

[5] A.K. Alves de Medeiros, B.F. van Dongen, W.M.P. van der Aalst, and A.J.M.M. Weijters. Process Mining: Extending the α-algorithm to Mine Short Loops. BETA Working Paper Series, WP 113, Eindhoven University of Technology, Eindhoven, 2004.

[6] F. Baccelli and P. Konstantopoulos. Estimates of cycle times in stochastic Petri nets.

[7] P. Backus, M. Janakiram, S. Mowzoon, C. Runger, A. Bhargava, N.N. America, and CA Gardenia. Factory cycle-time prediction with a data-mining approach. *Semiconductor Manufacturing, IEEE Transactions on*, 19(2):252–258, 2006.

[8] R.A. Becker. A Brief History of S. *Computational Statistics*, pages 81–110, 1994.

[9] C.F. Chien, C.W. Hsiao, C. Meng, K.T. Hong, and S.T. Wang. Cycle time prediction and control based on production line status and manufacturing data mining. *Semiconductor Manufacturing, 2005. ISSM 2005, IEEE International Symposium on*, pages 327–330, 2005.

[10] A.K.A. de Medeiros and CW Gunther. Process mining: Using CPN tools to create test logs for mining algorithms. *system*, 15(05):00, 2004.

[11] A.K.A. de Medeiros and A.J.M.M. Weijters. ProM Tutorial.

[12] S.N. den Hertog. Case prediction in BPM systems: Research to the predictability of the remaining time of individual cases. Master's thesis, Eindhoven University of Technology, Eindhoven, 2008.

[13] J. Dippon, P. Fritz, and M. Kohler. A statistical approach to case based reasoning, with application to breast cancer data. *Computational Statistics and Data Analysis*, 40(3):579–602, 2002.

[14] B.F. van Dongen, A.K. Alves de Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, and W.M.P. van der Aalst. The ProM framework: A new era in process mining tool support. In *Application and Theory of Petri Nets 2005*, volume 3536 of *lncs*, pages 444–454. Springer, 2005.

[15] B.F.v. Dongen, R.A. Crooy, and W.M.P.v.d.Aalst. Cycle time prediction: When will this case finally be finished? *CoopIS*, 2008.

[16] K. .R. Haberle and R. J. Graves. Cycle time estimation for printed circuit board assemblies. *IEEE Transactions on Elektronics Packaging Manufacturing*, 24(3):188–194, July 2001.

[17] W. Hardle. *Applied Nonparametric Regression*. Cambridge University Press Cambridge, 1990.

[18] W. Hardle and J.S. Marron. Optimal bandwidth selection in nonparametric regression function estimation. *Annals of Statistics*, 13(4):1465–1481, 1985.

[19] T. Hayfield and J. S. Racine. *np: Nonparametric kernel smoothing methods for mixed datatypes*, 2008. R package version 0.14-3.

[20] K. Hornik. The R FAQ, 2008. ISBN 3-900051-08-9.

[21] C.M. Hurvich, J.S. Simonoff, and C.L. Tsai. Smoothing parameter selection in nonparametric regression using an improved Akaike information criterion. *Journal of the Royal Statistical Society: Series B: Statistical Methodology*, 60(2):271–293, 1998.

[22] Q. Li and J.S. Racine. Nonparametric estimation of conditional CDF and quantile functions with mixed categorical and continuous data. *Journal of Business and Economic Statistics, forthcoming*, 2004.

[23] A.K.A. de Medeiros. *Genetic Process Mining*. PhD thesis, Eindhoven University of Technology, Eindhoven, 2006.

[24] E.A. Nadaraya. On Estimating Regression. *Theory of Probability and its Applications*, 9:141, 1964.

[25] J. Racine and Q. Li. Nonparametric estimation of regression functions with both categorical and continuous data. *Journal of Econometrics*, 119(1):99–130, 2004.

[26] J.S. Racine, J. Hart, and Q. Li. Testing the Significance of Categorical Predictor Variables in Nonparametric Regression Models. *Econometric Reviews*, 25(4):523–544, 2006.

[27] H. A. Reijers. Case prediction in BPM systems: A research challenge. *Journal of the Korean Institute of Industrial Engineers*, 33:1–10, 2006.

[28] A. Rozinat, M. Wynn, W.M.P. van der Aalst, A.H.M. ter Hofstede, and C. Fidge. Workflow Simulation for Operational Decision Support using YAWL and ProM. *BPM Center Report BPM-08-04*, 2008.

[29] B. Schellekens. 2G Case Prediction in Staffware, 2005. Draft version of master's thesis, Eindhoven University of Technology.

[30] M.H. Schonenberg, B. Weber, B.F. van Dongen, and W.M.P. van der Aalst. Supporting Flexible Processes Through Recommendations Based on History. *Lecture Notes in Computer Science*, 5240:51–66, 2008.

[31] M. Smith and R. Kohn. Nonparametric regression using Bayesian variable selection. *Journal of Econometrics*, 75(2):317–343, 1996.

[32] TIBCO Software Inc. *Staffware Process Objects (SPO) Programmer's Guide*, 2004.

[33] P. Vieu. Nonparametric regression: Optimal local bandwidth choice. *Journal of the Royal Statistical Society. Series B (Methodological)*, 53(2):453–464, 1991.

[34] G.S. Watson. Smooth Regression Analysis. *Sankhya: The Indian Journal of Statistics*, 26:359–372, 1964.

[35] Process Mining Website. http://www.processmining.org.

9 Appendix A : Experiment setup

The experiment was done in ProM, the process mining framework, using a log of the process appealing to the tax on houses of a municipality. This log is not publicly available, however more information can be found in [23].

Initially the log is loaded into ProM, as shown in Figure 44, so we can start to analyze it. ProM offers a great deal of plugins with which it is possible to analyze a log.

Figure 44: Screenshot of ProM with the log loaded

For the purpose of our experiments we must load the prediction miner plugin, in order to start our experiments. The initial screen of the prediction miner plugin is shown in Figure 45. The screenshot shows that we have already connected to 4 servers running Rserve, these servers will perform the calculations.

The second part the prediction miner plugin, shown in Figure 46, shows the settings of the plugin. These settings control all aspects of the regression performed. We can select to use cross validation, or to train only once on the log. For this experiment we will use the following settings.

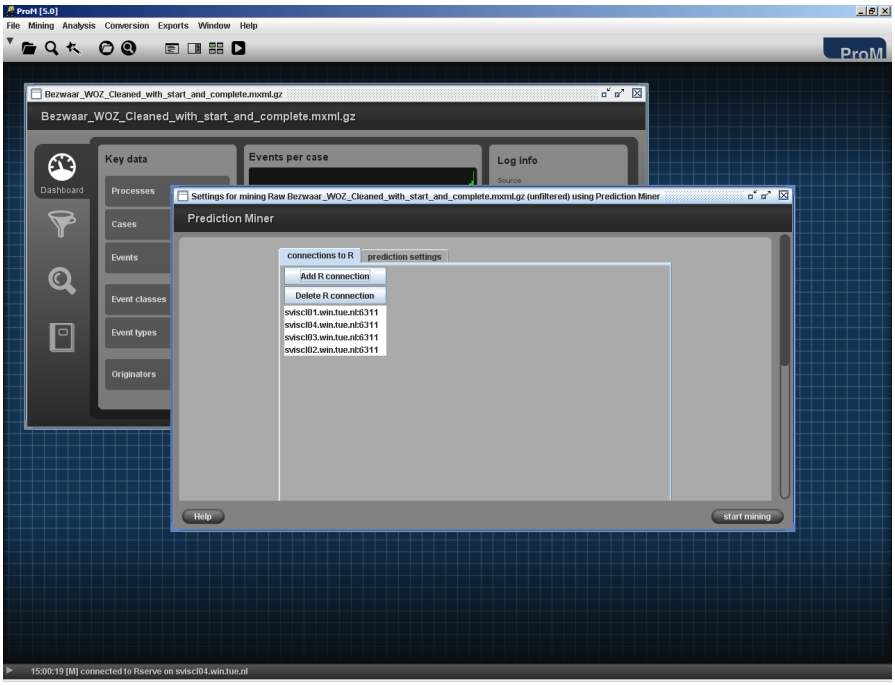


Figure 45: Screenshot of ProM with prediction plugin

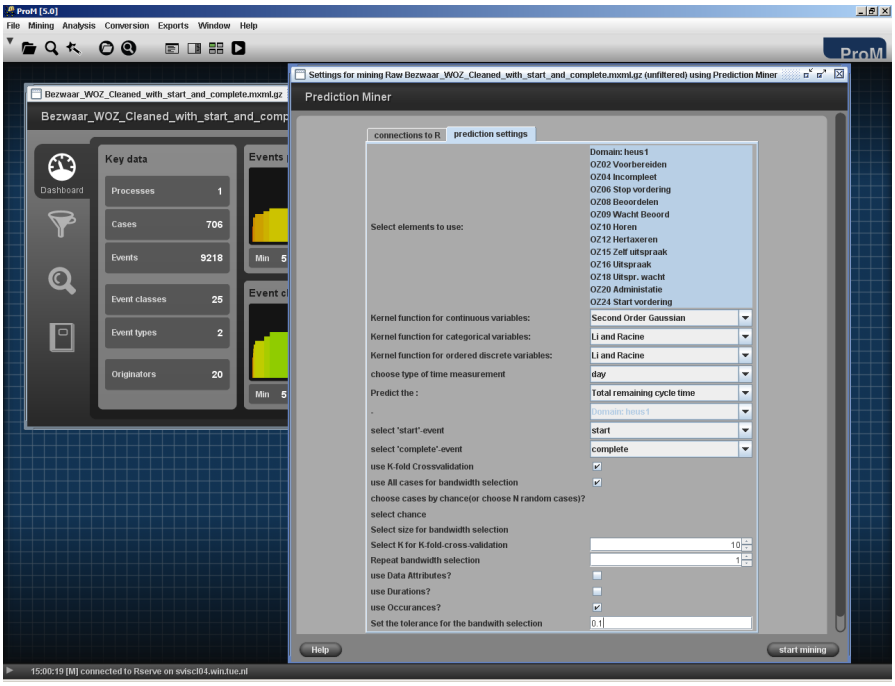


Figure 46: Screenshot of ProM with prediction plugin(2)

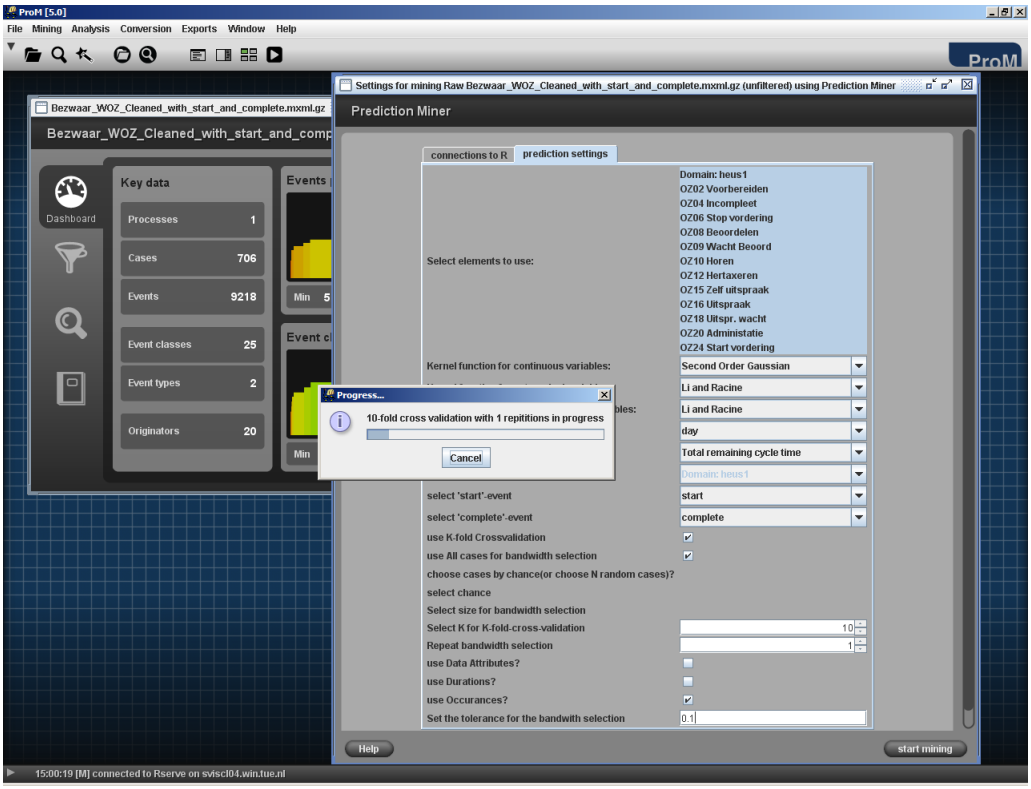


Figure 47: Screenshot of ProM with prediction plugin running

Running the prediction miner will, depending on the settings and the servers, take a long time. The settings as shown in the screenshot will result in the prediction of the total remaining cycle time with the non-parametric predictor based on occurrences. Eventually the plugin will finish, and Figure 48 will be shown. The results are summed up in a window with a tabbed view with tables, also the results were written into the log which is presented to the ProM framework again in order to further analyze it.

Table 20: Overview Settings prediction plugin

Setting	Options	Value for the experiments
elements to use	<activities in the process>	all activities
kernel function for continuous values	[Second Order Gaussian, Second Order Epachanikov, Uniform]	Second Order Gaussian
kernel function for categorical values	[Aitchison and Aitken, Li and Racine]	Li and Racine
kernel function for ordered discrete values	[Wang and van Ryzin, Li and Racine]	Li and Racine
type of time measurement	[second, minute, hour, day]	day
predict the	[total remaining cycle time, remaining cycle time until element <..>, occurrence of element <..>]	total remaining cycle time
start event	[start]	start
complete event	[complete]	complete
use k-fold cross-validation	[true,false]	true
use all cases for bandwidth selection	[true,false]	true
choose cases by chance	[true,false]	-
chance for the bandwidth selection	<number>	-
size for the bandwidth selection	<number>	-
k for k-fold cross-validation	<number>	10
repeat bandwidth selection	<number>	1
use data-attributes	[true,false]	true/false
use occurrences	[true,false]	true/false
use durations	[true,false]	true/false
tolerance for the bandwidth selection	number	0.1

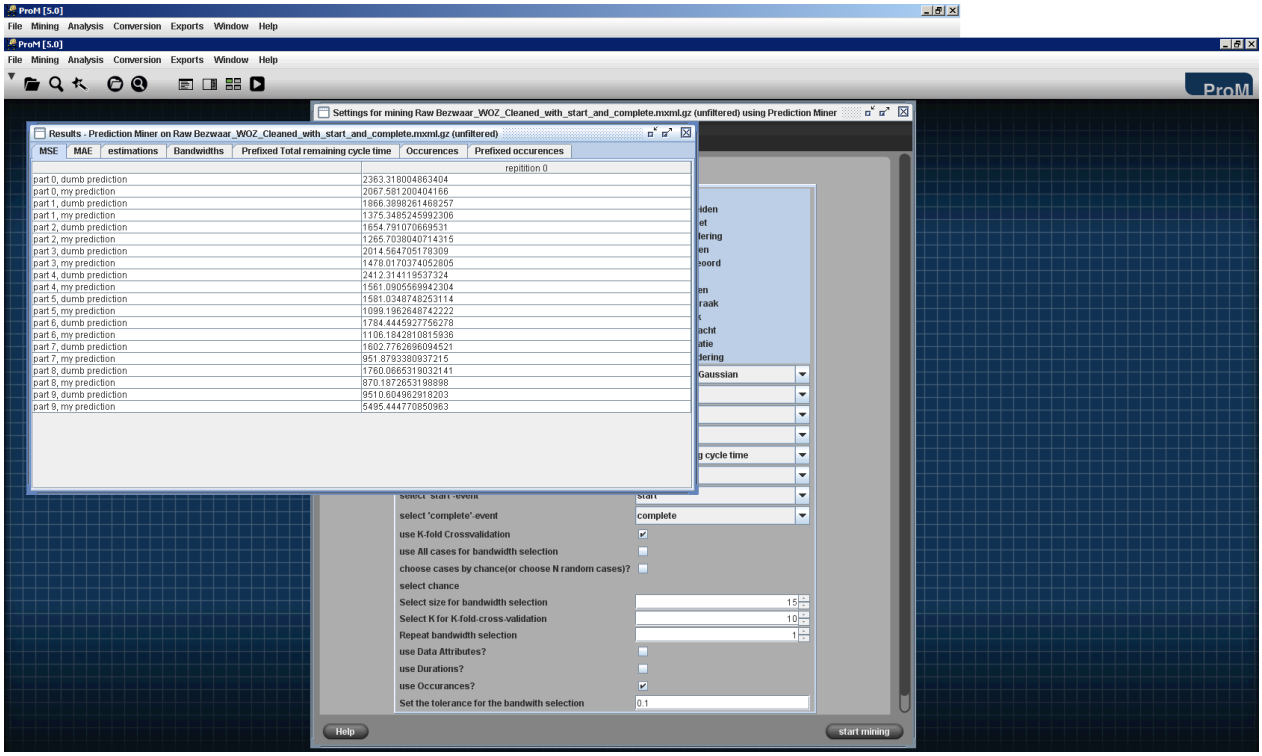


Figure 48: Screenshot of ProM with prediction plugin results

The table shows results such as the mean square errors/mean absolute error/ of each of the cross validated measurements, the actual estimations, and the varous measurements.

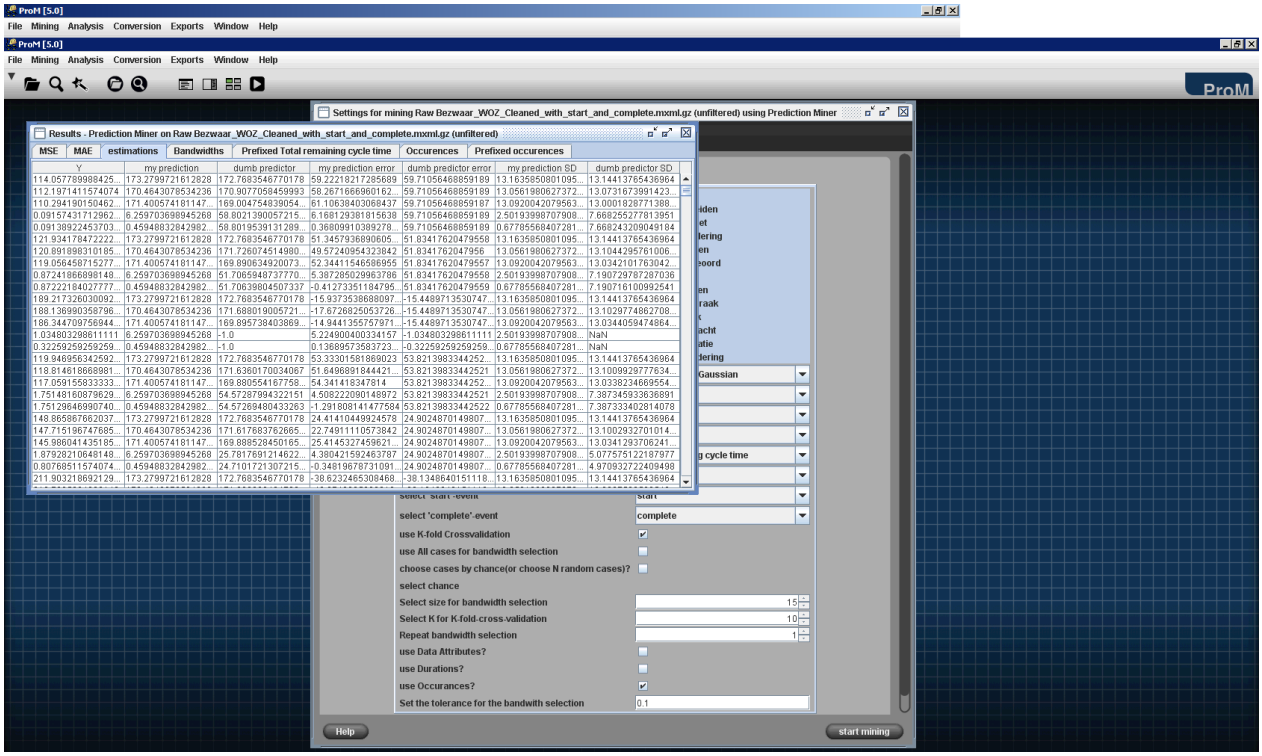


Figure 49: Screenshot of ProM with prediction plugin results

10 Appendix B : CPNTools Simulation models

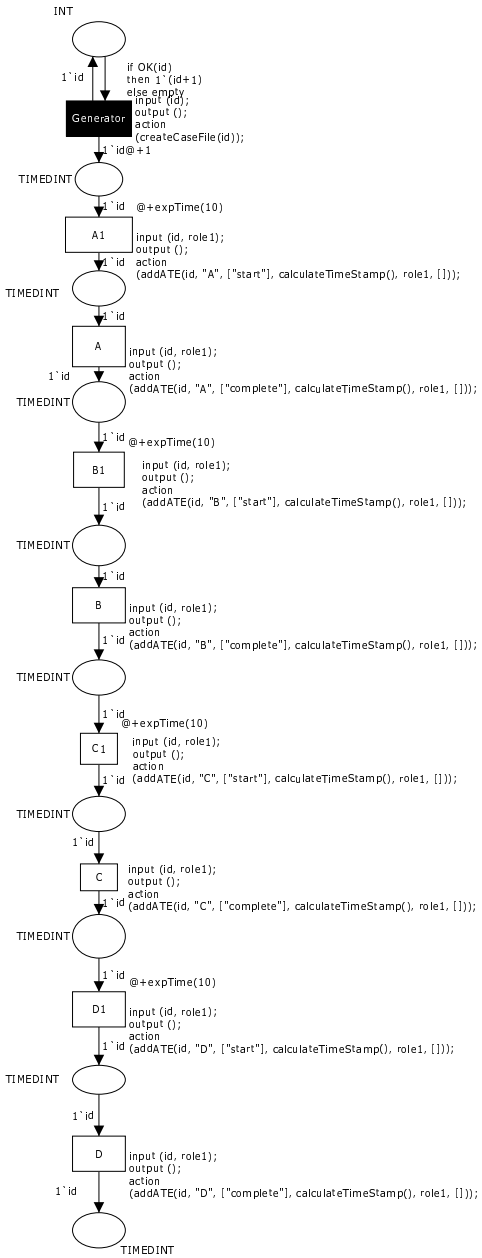


Figure 50: CPNTools simulation model 1a

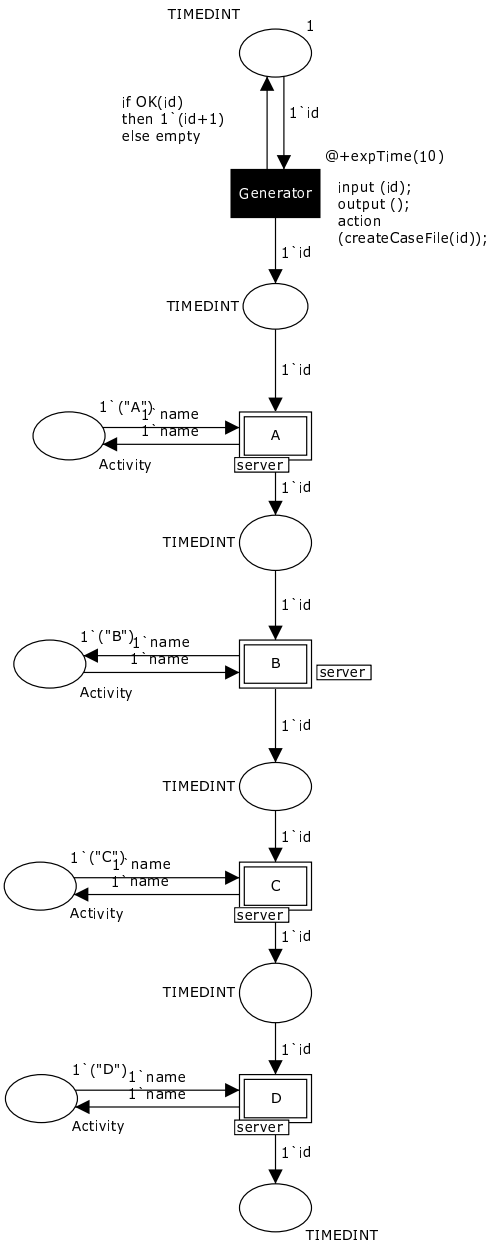


Figure 51: CPNTools simulation model 1b

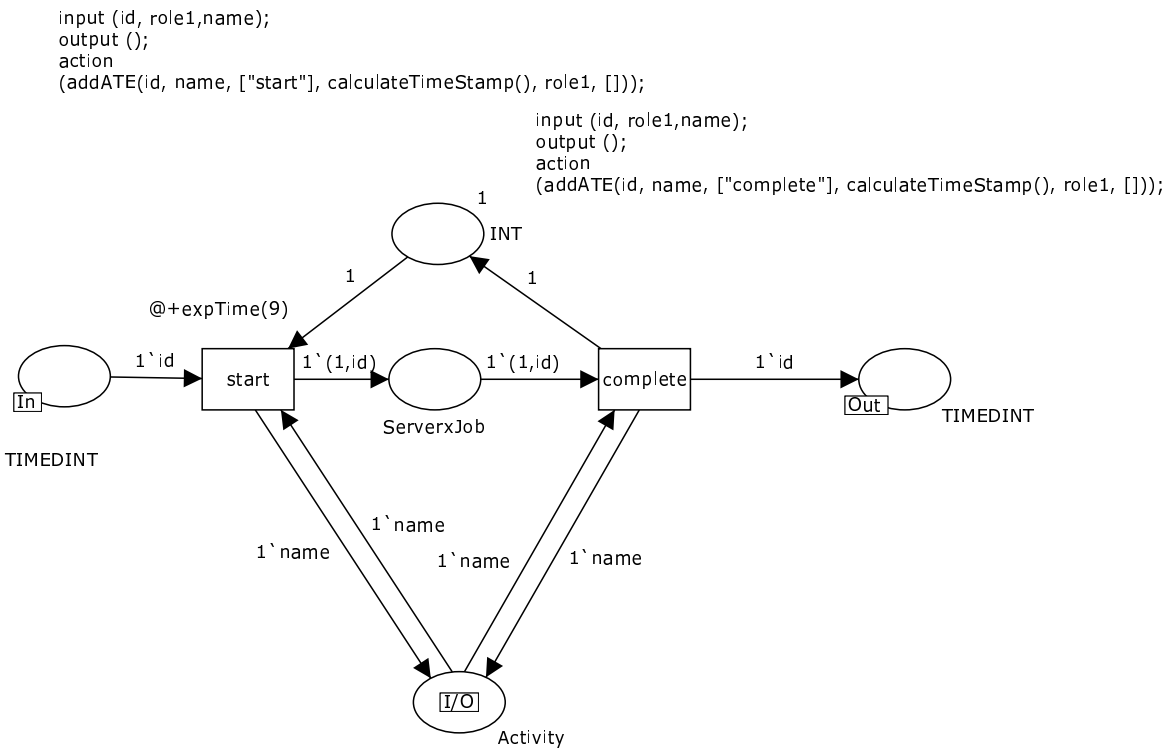


Figure 52: CPNTools simulation model "server"

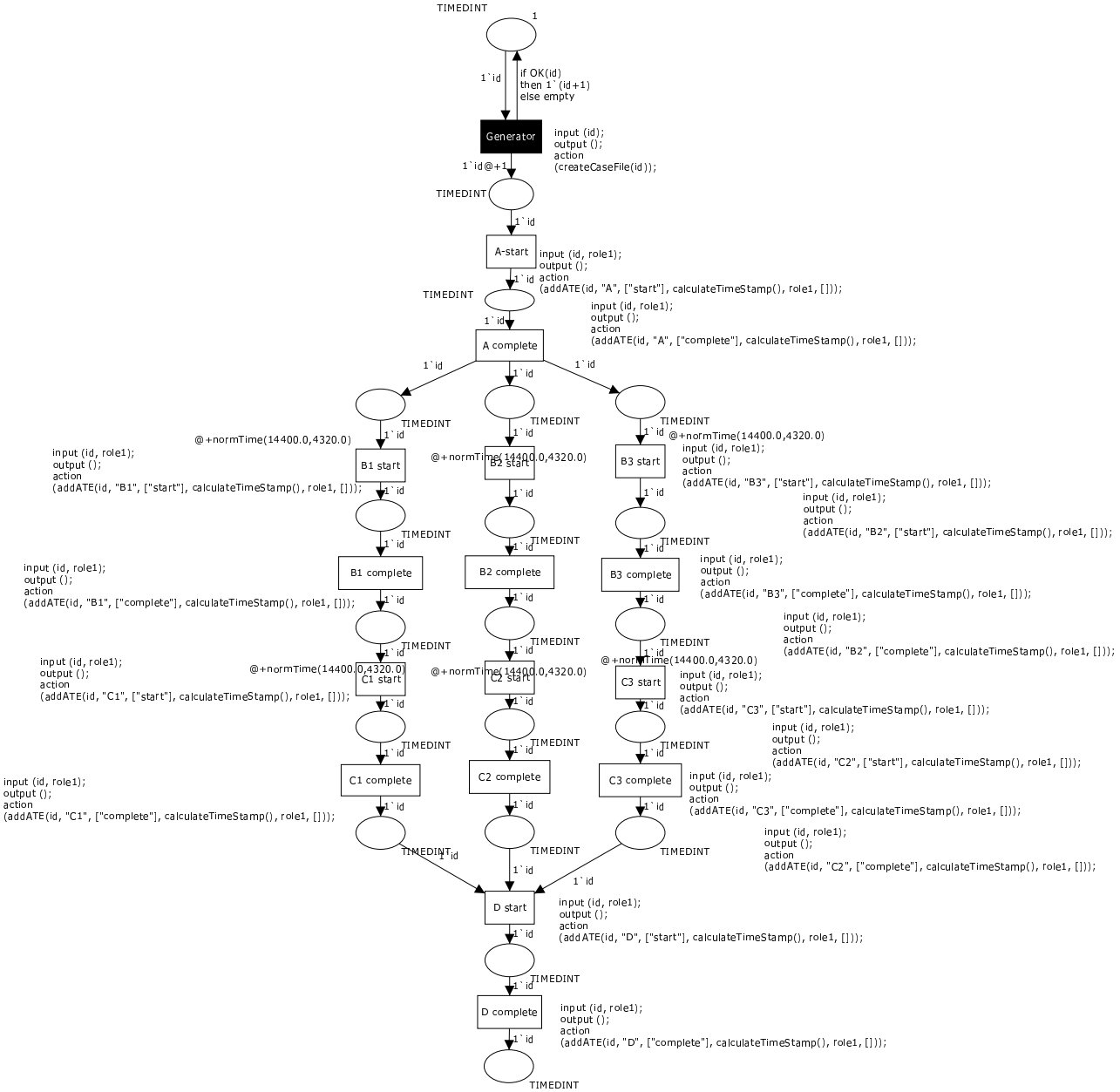


Figure 53: CPNTools simulation model 2

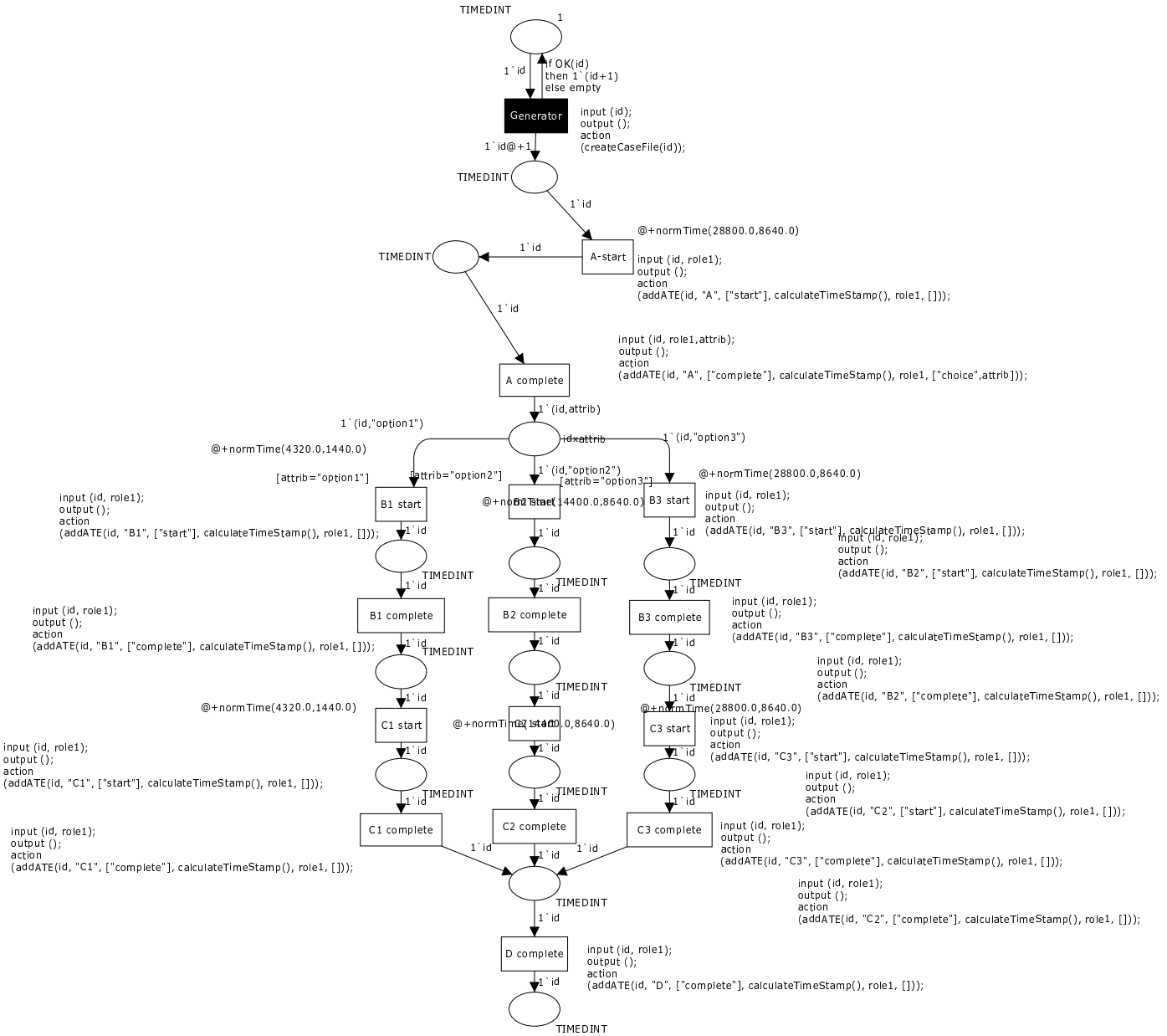


Figure 54: CPNTools simulation model 3

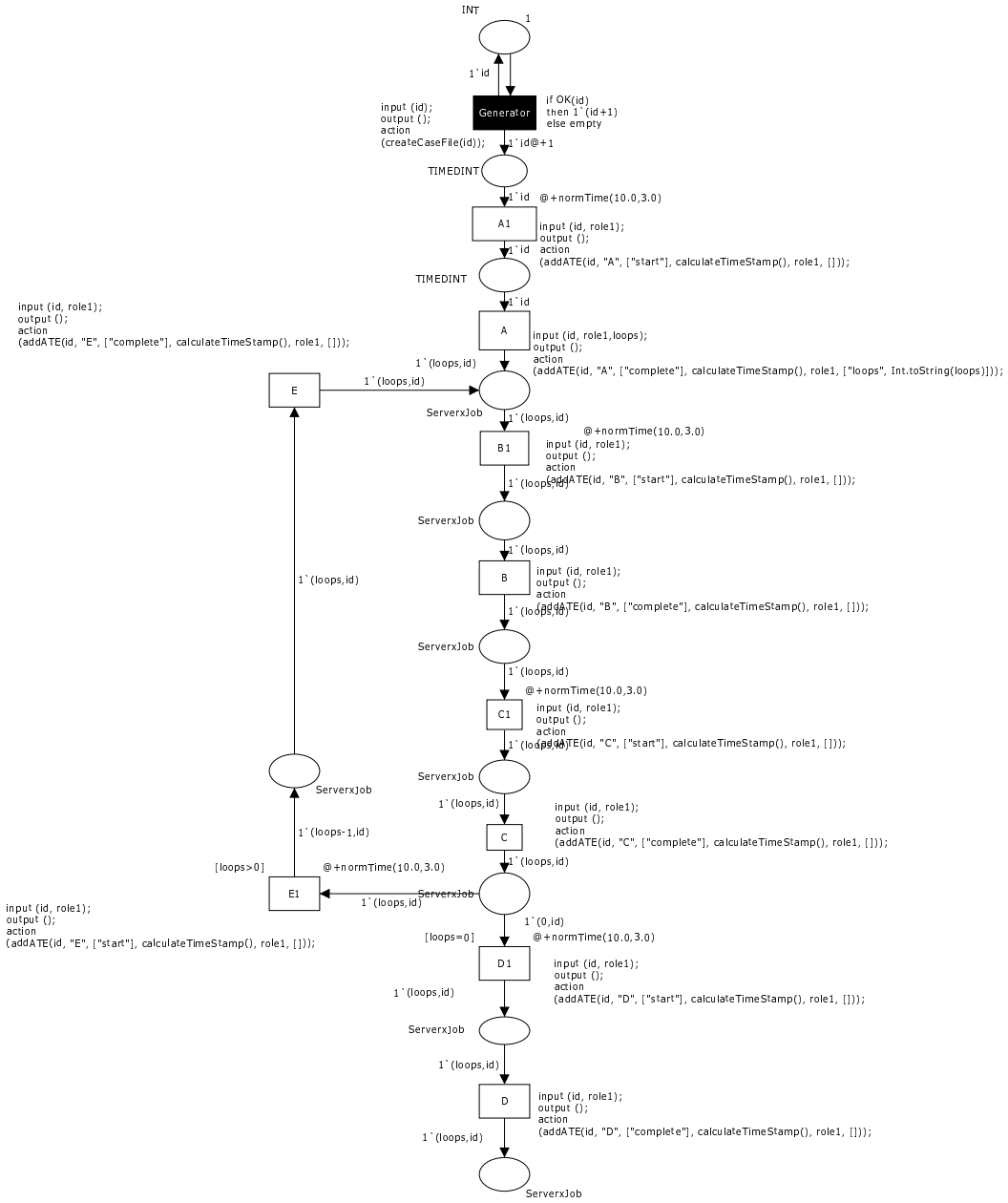


Figure 55: CPNTools simulation model 4a

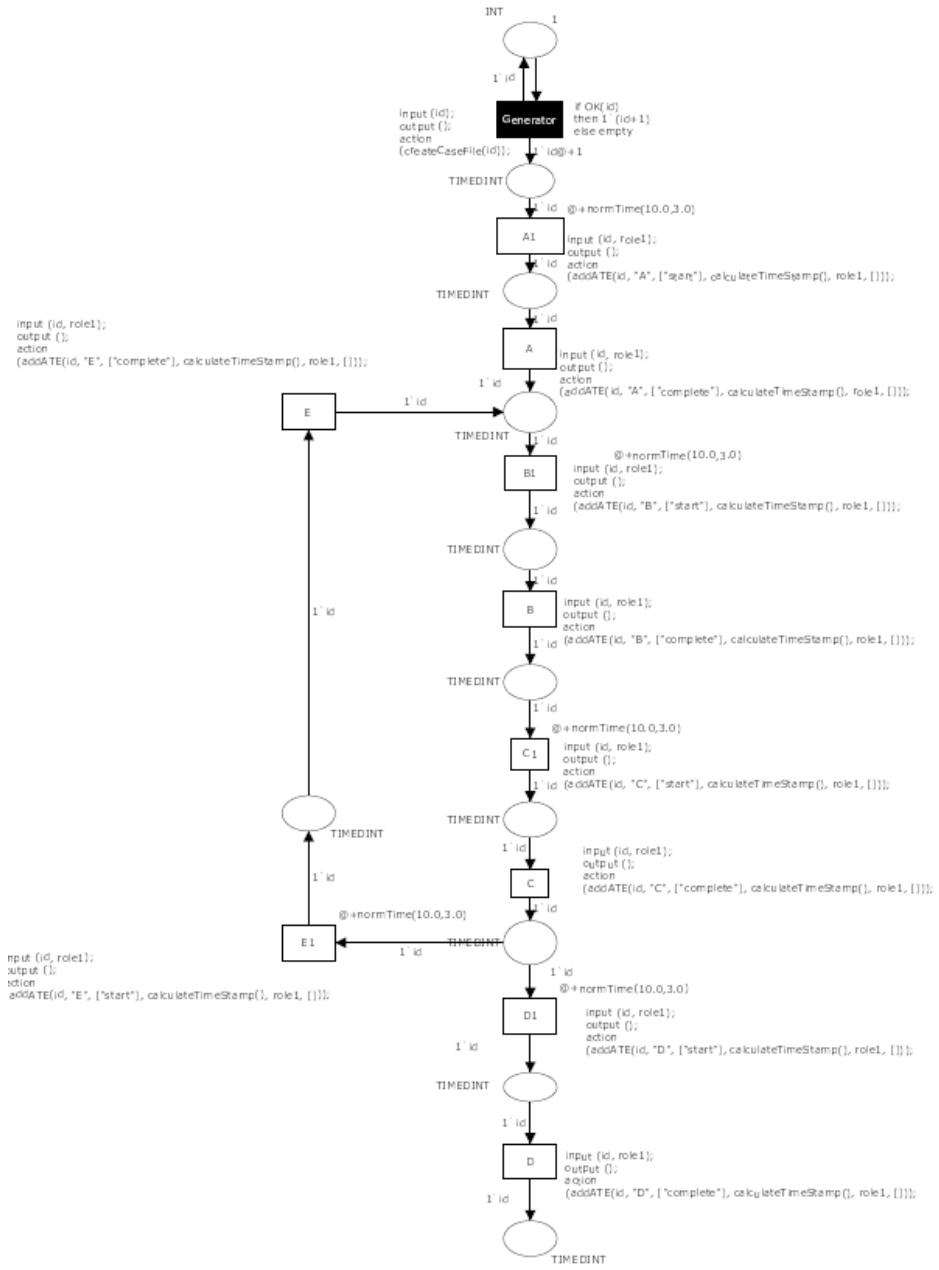


Figure 56: CPNTools simulation model 4b

11 Appendix C : Process Model based upon the log of the Case Study

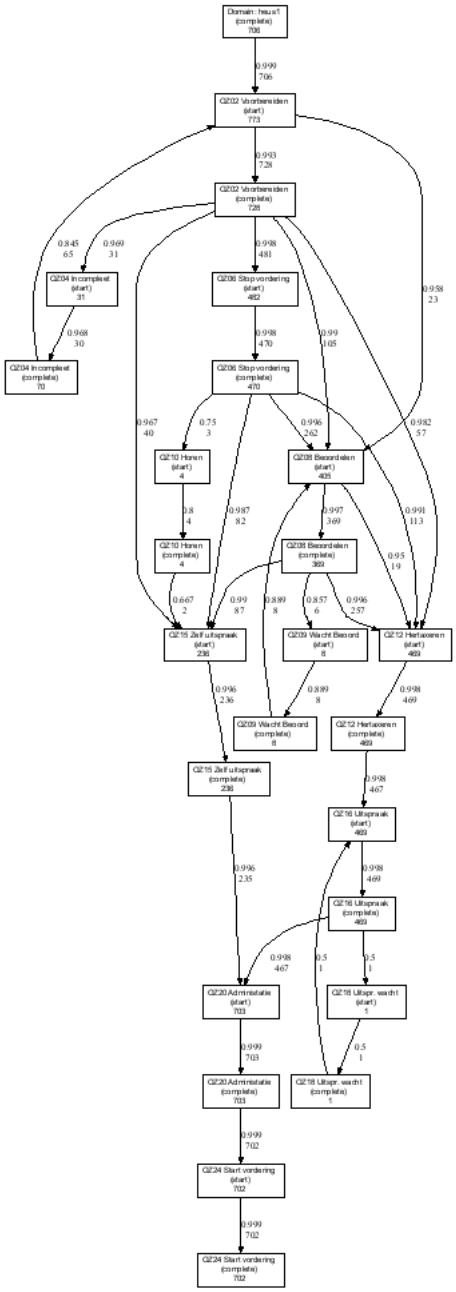


Figure 57: Process Model of the Case Study

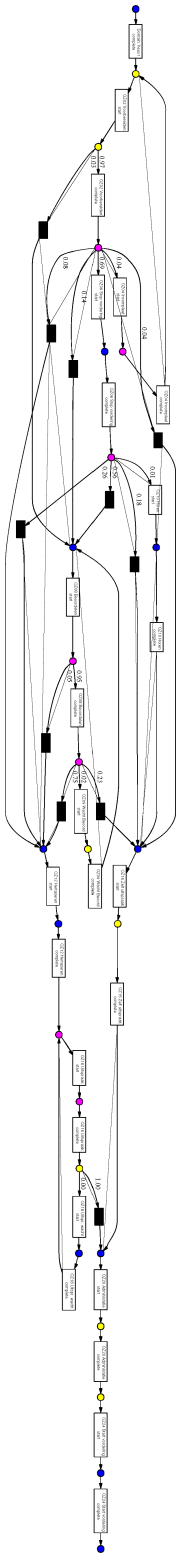


Figure 58: Petri Net with performance indication of the Case Study

12 Appendix D : Bandwidth values of the nonparametric prediction of the total remaining cycle time using attributes of the Case Study

Table 21: Bandwidth values of the non-parametric prediction of the total remaining cycle time using attributes of the Case Study

new_queue	o_resource	type	error	id	rule	queue	name	priority	time_stamp	workset
0,01149	0,00026	0,33895	0,99997	0,00449	0,99876	0,08552	0,01381	0,00137	0,76644	0,06708
0,02623	0,36730	0,89060	0,99998	0,00195	0,99762	0,13132	0,04941	0,04368	0,33677	0,13964
0,02772	0,46843	0,86008	0,99998	0,00238	0,99847	0,15544	0,04758	0,04268	0,51585	0,13706
0,02384	0,91299	0,99676	0,99994	0,00731	0,99940	0,15188	0,01740	0,05108	0,51242	0,16722
0,00622	0,38815	0,97134	0,98739	0,00968	0,99998	0,61930	0,00696	0,02598	0,40198	0,09201
0,02751	0,40038	0,89018	0,99987	0,00282	0,99641	0,08884	0,03269	0,04276	0,57884	0,18723
0,01107	0,77150	0,99954	0,98083	0,00857	0,99916	0,22329	0,01228	0,03743	0,21267	0,19911
0,02759	0,40346	0,89009	0,99998	0,00498	0,99687	0,14982	0,01439	0,04011	0,77040	0,09705
0,02352	0,36085	0,88451	0,00621	0,00484	0,99995	0,08707	0,01043	0,03061	0,34142	0,12497
0,02341	0,78296	0,91297	0,98714	0,00043	0,99251	0,08713	0,16666	0,02213	0,99053	0,12134

The rows in this table represent the 10 bandwidths as calculted by the 10-fold cross-validation.

13 Appendix E : Bandwidth values of the nonparametric prediction of the occurrence of "0Z12 Hertaxeren" using occurrences of the Case Study

The rows in this table represent the 10 bandwidths as calculted by the 10-fold cross-validation.

Table 22: Bandwidth values of the nonparametric prediction of the occurrence of "0Z12 Hertaxeren" using occurrences of the Case Study

oz20	0z18	0z12	0z10	0z16	0z15	0z02	0z04	0z09	0z06	0z08		0z24
administratie	uitsprwacht	hertaxeren	horen	uitspraak	zelfuitsprk	voorbereidn	incompleet	wachtbeoord	stopvorderng	beoordelen	domainheus1	startvorderng
2.35E-04	0.996845	2.71E-04	8.31E-04	5.74E-03	0.622222	0.999999	3.81E-03	0.002427	8.89E-01	0.460793	0.016886	0.999548
0.001426	0.094331	2.40E-06	8.31E-04	0.009928	0.986649	0.999256	0.003501	0.003375	0.997967	0.460793	0.022563	0.999992
0.001495	0.093431	1.05E-06	0.001301	0.008736	0.998055	0.9999	0.003956	1.26E-04	0.968292	0.460793	0.021737	0.999995
9.90E-04	0.999258	1.66E-06	0.003174	0.998612	0.989352	0.145888	0.004733	0.011913	9.29E-04	0.460793	0.014047	0.998103
3.37E-04	0.096817	3.54E-06	0.001463	0.008615	0.978289	0.999256	0.004296	0.002062	0.453143	0.460793	0.017699	0.999987
0.001333	0.091349	2.19E-06	5.81E-06	0.006278	0.992813	0.992355	0.003451	0.00369	0.999691	0.460793	0.017009	0.999997
0.001147	0.277147	3.13E-05	7.34E-04	0.008777	0.292017	0.999943	0.004171	0.002537	0.969115	0.460793	0.022575	0.999492
0.001081	0.998818	3.86E-05	0.002964	0.005481	0.977255	0.999981	0.003669	0.002721	0.995753	0.460793	0.018954	0.999336
7.12E-04	0.346665	1.10E-05	8.67E-04	6.30E-04	0.98658	0.999936	0.005277	0.002554	0.99888	0.460793	0.006989	0.999999
0.001792	0.991992	4.56E-05	0.527785	0.987721	0.999944	0.189582	0.001348	1.15E-04	0.999864	0.460793	0.012711	0.994009

14 Appendix F : Bandwidth values of the nonparametric prediction of the occurrence of "0Z12 Hertaxeren" using attributes of the Case Study

Table 23: Bandwidth values of the nonparametric prediction of the occurrence of "0Z12 Hertaxeren" using attributes of the Case Study

new_queue	o_resource	type	error	id	rule_num	queue	name	priority	timestamp	workset
0.999893	0.964036	0.954539	0.025045	9.65E-18	0.064335	0.980805	2.18E-05	0.07529	0.098259	0.996528
0.999999	0.675157	0.901458	4.14E-04	1.86E-16	0.164746	0.621298	6.26E-08	0.116095	0.677209	0.999991
1	0.582591	0.982406	5.85E-04	2.49E-18	0.107479	0.998514	9.40E-08	0.044906	0.80048	0.999821
0.999992	0.426587	0.999999	7.75E-14	1.77E-17	0.105036	0.999926	1.51E-07	0.033186	0.3141	0.999995
1	0.91454	0.996671	3.68E-05	7.99E-17	0.10341	0.802329	1.27E-09	0.159488	0.50883	0.999995
0.999999	0.999996	1	7.01E-04	3.01E-17	0.040972	0.974106	6.48E-08	6.08E-04	0.903802	0.999998
1	0.992911	0.820255	0.34045	2.44E-19	0.155134	0.881992	2.90E-10	2.94E-06	0.390378	1
0.794306	0.485053	0.99986	6.44E-05	8.38E-18	0.028888	0.563247	4.76E-10	0.024957	0.440867	0.999997
1	0.661551	0.8251	2.40E-06	5.88E-18	0.093585	0.999677	2.24E-11	0.098347	0.032173	0.999907
0.999999	0.684836	0.991553	1.11E-04	3.75E-17	0.085861	0.960532	6.24E-09	0.181782	0.92384	0.999998

The rows in this table represent the 10 bandwidths as calculted by the 10-fold cross-validation.

15 Appendix G: The events for cases 1 to 4 of the running example

Timestamp	Event Type	Element	Attributes
01.01.1970 09:38:00.000	complete	Register	-
01.01.1970 09:38:00.000	start	Analyze Defect	-
01.01.1970 09:48:00.000	complete	Analyze Defect	defectType: 9 phoneType: T2
01.01.1970 10:04:00.000	start	Repair(Complex)	-
01.01.1970 10:18:00.000	complete	Repair(Complex)	-
01.01.1970 10:18:00.000	start	Test Repair	-
01.01.1970 10:28:00.000	complete	Test Repair	defectFixed: true numberRepairs: 0
01.01.1970 10:32:00.000	complete	Inform User	-
01.01.1970 10:35:00.000	complete	Archive Repair	defectFixed: true numberRepairs: 0

Table 24: Events of Case 1

Timestamp	Event Type	Element	Attributes
02.01.1970 15:58:00.000	complete	Register	-
02.01.1970 15:58:00.000	start	Analyze Defect	-
01.01.1970 16:03:00.000	complete	Analyze Defect	defectType: 8 phoneType: T2
02.01.1970 16:18:00.000	complete	Inform User	-
02.01.1970 16:22:00.000	start	Repair(Complex)	-
02.01.1970 17:06:00.000	complete	Repair(Complex)	-
02.01.1970 17:06:00.000	start	Test Repair	-
02.01.1970 17:13:00.000	complete	Test Repair	defectFixed: true numberRepairs: 0
02.01.1970 17:23:00.000	complete	Archive Repair	defectFixed: true numberRepairs: 0

Table 25: Events of Case 2

Timestamp	Event Type	Element	Attributes
03.01.1970 10:10:00.000	complete	Register	-
03.01.1970 10:10:00.000	start	Analyze Defect	-
03.01.1970 10:20:00.000	complete	Analyze Defect	defectType: 6 phoneType: T1
03.01.1970 10:43:00.000	start	Repair(Complex)	-
03.01.1970 10:56:00.000	complete	Repair(Complex)	-
03.01.1970 10:56:00.000	start	Test Repair	-
03.01.1970 11:03:00.000	complete	Test Repair	defectFixed: true numberRepairs: 0
03.01.1970 11:06:00.000	complete	Inform User	-
03.01.1970 11:09:00.000	complete	Archive Repair	defectFixed: true numberRepairs: 0

Table 26: Events of Case 3

Timestamp	Event Type	Element	Attributes
04.01.1970 09:02:00.000	complete	Register	-
04.01.1970 09:02:00.000	start	Analyze Defect	-
04.01.1970 09:09:00.000	complete	Analyze Defect	defectType: 1 phoneType: T1
04.01.1970 09:14:00.000	start	Repair(Simple)	-
04.01.1970 09:19:00.000	complete	Repair(Simple)	-
04.01.1970 09:19:00.000	start	Test Repair	-
04.01.1970 09:28:00.000	complete	Test Repair	defectFixed: true numberRepairs: 1
01.01.1970 09:32:00.000	complete	Inform User	-
01.01.1970 09:32:00.000	complete	Archive Repair	defectFixed: true numberRepairs: 1

Table 27: Events of Case 4