universität
wien

# MASTERARBEIT

Titel der Masterarbeit

„Evaluation and Implementation of a Visualization Component for Difference Analysis of Process Models“


verfasst von

Manuel Gall BSc


angestrebter akademischer Grad

Diplom-Ingenieur (Dipl.-Ing.)

Wien, 2015

| | |
|---|---|
| Studienkennzahl lt. Studienblatt: | A 066 926 |
| Studienrichtung lt. Studienblatt: | Masterstudium Wirtschaftsinformatik |
| Betreut von: | Univ.-Prof. Dipl.-Math. Dr. Stefanie Rinderle-Ma |

## Acknowledgments

I´d like to thank my supervisor, Stefanie Rinderle-Ma, for the patience and continuously support through the whole work.

Thanks to Simone Kriglstein for the discussions and advices in terms of visualization. As well as the incredible guidance and support through the survey design and evaluation.

Thanks to my parents Angelika and Siegfried and my brother Benjamin for the encouragements and taking my mind off work - to start fresh again.

# Contents

# 1 Introduction

With the use of information systems like Enterprise Resource Planning (ERP), Workflow-Management-Systeme (WfMS), Customer Relationship Management (CRM) and Supply Chain Management (SCM) processes are executed, monitored and data is generated. Process mining tools like ARIS, Disco, Celonis business intelligence, ProM, Rbminer/Dbminer and many more are able to process these data sets and offer various application areas like mining, conformance and extension.

The key area of application is the discovery/mining of process models based on the data provided by the information system. The purpose of the newly discovered process model is analysis and optimization. Analysis includes visualization of control- and data-flow as well as finding bottlenecks, run-time analysis, determine involved actors and many more [27, 32]. The other important area of application which relates to this thesis is conformance checking. Conformance checks compare a generated process model (Actual-Model) with an ideal process model (Target-Model). This comparison helps to determine where the differences between the two processes lay. Detecting differences leads to adaption and/or rethinking [2]. Adaption means to adapt the real world process in order to converge to the target-model. Rethinking means to review the ideal process maybe it is not ideal.

By now the purpose of conformance checks is to compare generated process models with ideal process models. Modern organizations need to adapt their processes to secure their business. To do so they need to compare real world processes. For example, comparing the production of the same good in two different facilities can show why one facility is faster in producing the good then the other. Besides comparing two different real world processes comparing one process at different points in time could also be from interest. An example would be a facility which produces one type of good over the period from 2013 to 2014. Comparing the years 2013 and 2014 can lead to an answer why in 2013 the good was produced 5 percent faster.

When such differences between two processes are calculated it is necessary to visualize the information in a way the user can understand where the processes differ from each other. The gained knowledge obtained by this comparison can be used to analyze, optimize and merge two processes.

### 1.1 Contribution and Research Methodology

This thesis contributes to previous work in the field of process mining. The thesis builds on the concept introduced by [43], whose goal was to develop a concept and a visualization approach for the difference analysis. Based on this concept a literature research targeting visualization approaches was performed. These visualization approaches are part of a survey. The result of the survey will affect the prototype implementation. The goal of this thesis is to implement a prototype which visualizes the difference between two processes. The following is considered to be contributed in this work:

- Based on a literature research several visualizations how the difference can be visualized are shown. To secure the best visualization is chosen an evaluation will be conducted.
- Combining the concept and evaluation results in a way to be able to build a prototype. The prototype will use already implemented mining algorithms for process model generation. These models are then used for difference calculation and visualization.

### 1.2 Structure of Thesis

The thesis is structured as followed, Section 2 gives a brief introduction in process mining to understand the fundamentals. Section 3 introduces the difference graph concept based on [43]. Based upon this concept Section 4 discusses several ways how the difference graph can be visualized. In order to find the best suiting representation an evaluation was conducted which is part of Section 6. Section 7 and 8 cover the implementation process of the prototype which brings together the concept and the evaluation. Sections 9 and 10 wrap up the thesis by summarizing what was achieved and which future research can be build up on this thesis.

### 1.3 Related Work and Overview

Building and visualization of a difference graph is an interdisciplinary field. It uses knowledge from visualization [41, 25, 6, 37, 38], business processes [1, 55, 2, 57, 32, 27] and analysis [53, 34] . In order to build a difference graph, first a process model [43, 2] is needed. It is generated through process mining techniques [12, 4, 5, 18] which combines the knowledge from business processes and analysis. How such a model is generated and which information is necessary to build a process model is part of Section 2.

Building the difference between two graphical representations has been in research for years [3, 13]. Directly building the difference of process models generated from instance traffic is addressed in [53]. The author states that his approach uses the instance traffic to calculate more and less important parts of two process models and expresses their equality by a value between 0 and 1. The approaches have in common that they map the difference to one single value.

With only one value it can not be understood where the differences between two process models can be found. The only knowledge available is that there is a difference. A graphical visualization where the differences are calculated for each node and edge can show exactly where two models are different.

To be able to build a graphical visualization for the difference graph knowledge from the fields business processes and visualization has to be combined. The fields have a long history together, one of the first visualization approaches were Petri Nets [55]. Since then many different visualization languages for business processes have arose [44, 50, 54]. A rather uncommon type of visualization is 3D [38, 37] visualization. But so far no paper addresses the discussion if a 2D or 3D representation should be preferred for visualizing a difference graph. This topic is addressed in Section 4.1.

A combined approach for the three fields visualization, business processes and analysis is presented in [43]. The authors give an introduction how the difference of two process models generated from instance traffic can be calculated and visualized. The authors showed a visualization approach but did not evaluate their approach. To extend their work Section 4.2 covers a literature research with the goal to find the best suiting visualization for the difference of two process models. The research showed many possible ways how the difference can be visualized. An evaluation was conducted to determine which of the styles stated in Section 4.3 should be used to visualize the difference between two process models. Afterwards an implementation of the difference graph concept, shown in Section 8, is performed.

## 2 Fundamentals of Process Mining

The following sections will give an introduction where the information to build a process model can be gathered from and how this information is transformed into a process model through process mining algorithms. To distinguish between these algorithms three categories will be presented. Afterwards four quality criteria will be introduced to analyze the output quality of the algorithms. This quality criteria will be discussed and should show if there is one specific category or algorithm which has to be considered for the difference calculation.

### 2.1 From a Log to a Model

Computer driven work and usage of information systems is an indispensable part of modern business. This leads to a wealth of data which companies generate. To support decision making it is important to gain knowledge from this data. Data mining is a process where knowledge is extracted from data. It can be distinguished into descriptive mining which allows to characterize the data and predictive mining to allow predictions based on the data. The extracted knowledge should explain the data for example through decision rules or decision trees. In order to analyze processes it is no suitable technology because the structural dependencies are the relevant information which processes contain. Derived from data

mining was process mining which also extracts structural patterns with the focus on discovery of structural models for example process models.

Process models visualize the dependencies between activities of business processes. There are many different process modeling languages to represent these process models for example Business Process Model and Notation (BPMN), Event-driven Process Chains (EPC) or Petri Nets. To be able to support a wide area of application the difference analysis does not require a specific modeling language. It only needs a process model. This work is built on the process model definition from [43]. They define a process model as a directed connected graph ($N$, $E$), where $N$ is defined as a set of nodes and $E$ a set of directed control edges. Each node consists of an unique id, label and node type. $E \subseteq N \times N$ is a set of directed edges between nodes. Some additional requirements are that there is one defined start node which has no incoming edges and one defined end node which has no outgoing edges. Furthermore each node has to be on a path between start node and end node. A simple process model is illustrated in Figure 1. It consists of four labeled nodes with weights and four weighted edges. The edge from node „A“ to node „B“ has a weight of two while all other nodes have a weight of one. Node „A“ is the start of the process and it ends with node „D“, both have a weight of two. The weight expresses the instance traffic, more details about instance traffic can be found in Section 3.2. The process for this example was executed two times. The first execution comprised activities „A“, „B“, „C“, „D“ and the second activities „A“, „B“, „D“. In this example a weight of two represents two executions. Activities „A“, „B“, „D“ where executed two times and activity „C“ was executed once. Every edge consists of a weight, edges represent the execution order, for example, after activity „A“ was executed activity „B“ followed. The behavior that activity „B“ followed activity „A“ was observed two times therefore the weight for edge „A“ to „B“ is two.

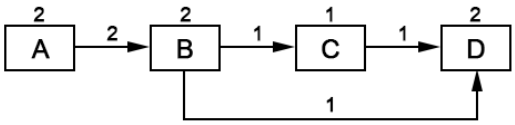



Figure 1: Simple process model with four activities and four edges. The weight for each activity and edge is visualized above of them.

The information which activities where executed in which order is stored in log files. These logs can be extracted from tools like Workflow-Management-Systems (WMS), Enterprise-Resource-Planning-Systems (ERP) or Costumer-Relationship-Management-System (CRM). Even without such tools it is possible to generate a log because a minimum log only consists of two components, i.e., activities and traces [55, 58]. Table 1 shows a minimal log. An activity cor-

responds to an execution step in a process and a trace is an identifier for a set of chronologically ordered activities obtained from one process instance. Table 1 shows the corresponding process log from which Figure 1 was generated. Trace 1 shows an instance which was started by executing activity „A“ after that activities „B“, „C“ where executed and it was completed with „D“. These traces are summarized to build a process log. This log describes why the weight for the edge from node „A“ to node „B“ is 2. There are two entries in the log for this edge. Trace 1 has an entry for activity „A“ and then activity „B“ is executed trace 2 shows the same order for activity „A“ and „B“. An important thing to notice is that trace 2 started before trace 1 was completed. This is no problem. It would also be possible that trace 2 starts and ends before trace 1 the order of traces is not important for generating a process model. It is required that for each trace the activities are in the order of execution. For example if trace 1 activity „A“ and trace 1 activity „B“ will swap places in the log but activity „A“ is still performed before activity „B“ it will be a violation and the log is inconsistent.

| Trace | Activity |
|---|---|
| 1 | A |
| 1 | B |
| 1 | C |
| 2 | A |
| 1 | D |
| 2 | B |
| 2 | D |

Table 1: This table shows a simple process log which corresponds to the process model shown in Figure 1. It consists of seven entries from two instances with four different activities.

In addition to activities and traces logs allow to store timestamps, role of execution and many more attributes. Additional data leads to additional knowledge which can be derived through process mining, for example with role of execution, relations between performers can be extracted by mining the organizational perspective. [57] A major advantage of process mining is that a log only needs activists and traces, it is not important how the log is obtained or from which system the data is extracted, the only thing that matters are activities and traces.

For the difference analysis a minimal log is sufficient because it is possible to generate a process model representing the log. To extend its usability the prototype implementation provided in this thesis will although require a timestamp. This is not considered as a problem because most of the available logs contain timestamps [45]. Timestamps allow to select instances specific to their

time of occurrence. For example a log contains traces from 2012 to 2014 and the difference analysis should be applied to analyze the difference between the years 2013 and 2014. One thing to consider is that the timestamp has to be obtained consistent before or after each activity. It is not suitable to record the timestamp sometimes before and sometimes after an activity because this will falsify the process model.

### 2.2 Differences of Mining Algorithms

It was mentioned before that the log is transformed into a process model therefore so called mining algorithms are needed. The procedure how the process model is generated is different for each mining algorithm, but they can be grouped into three categories [27, 2].

**Algorithmic techniques** produce on same input always the same output. This category was the first invented in the area of process mining [58]. Based on an algorithm the process model is generated step by step some major mining algorithms are the alpha miner [58], fuzzy miner [12] and heuristics miner [5].

**Genetic process mining** [4] uses an evolutionary approach which consists of several steps. The initialization step generates multiple random process models. A fitness function is applied to each model to measure how good it represents the log file. Followed by a selection of models for example only take the best fitting 50 percent. Afterwards genetic operations like crossover and mutation are used to generate new individuals (process models). They are the basis of the next generation if the termination criteria is not met, if the termination criteria is met a process model was found.

**Region based algorithms** [18, 7] transform the process log into a transition system and with the theory of regions a transition system is transformed into a Petri Net. A transition system describes the possible states the process can be in and the transitions between this states.

Each of these algorithms produces a process model. For this reason it is theoretically not necessary to use a specific algorithm for the difference analysis but there are some considerations stated in Section 7.2 which influence the choice of algorithms for the prototype implementation.

In order to preselect mining algorithms research was done but [2]

show that there is no perfect mining algorithm. They have analyzed different kinds of algorithms. Their example shows that in certain categories they achieve better and worse results.

From a process model it is expected that it is representative for the data stored within the log. But to achieve a good representation the mining algorithm has to solve a tradeoff between four quality criteria [27, 2]. The **fitness** criterion states if it is possible to retrace every trace from the log with the process model. **Precision** and **generalization** measure if the model is overfitting or underfitting. A model is called overfitting if it allows only the behavior stored within the log file this model has a good precision. Underfitting measures if the model allows additional behavior because the log might not be complete. Therefore it is more general. **Simplicity** or sometimes called **structure** measures if the model

makes good use of the vocabulary provided by the modeling language. A language often provides different ways to express the same behavior. One or some of this ways might be preferred over others because it is a „better“ representation [52]. This quality criteria show that it is always a balancing act that the mining algorithm tries to perform as good as possible.

On basis of this criteria a restriction to certain algorithms for the difference analysis is not possible, since any algorithm can provide the desired results under certain circumstances. Sometimes it is necessary to choose a higher degree of precision while other times the generalization stands in the foreground [27]. To know that these four criteria do not allow a restriction is very important in the later stage of this work. Section 7.2 covers the selection of mining algorithms for the prototype implementation.

### 2.3 Process Mining Applications

Creating process models, social networks or other organizational models with a mining algorithm which operates on data stored within log files is called process discovery. Process mining does also consist of two additional application areas conformance [51, 2] and extension [2].

If an organization already has process models, it can perform a **conformance check**. In order to do this a log of the corresponding process is needed and a model which was crafted by hand or some other process discovery method. To do a conformance check for each trace in the log it will be checked if the model expresses this behavior [33]. To get a sense if the model is a good representation of the log file the before mentioned quality criteria come in handy. With these criteria it is possible to understand where the model has a lack of information and what was represented very well. This information helps to understand differences between the model and the real world because real world processes are under continuous transformation. An example area of application is the combination of conformance checking and WFMS. The model which is used to execute the workflow can be extracted and checked if it is a good representation of the corresponding real world process. This helps to minimize deviations between WFMS and real world. [2]

**Extension** allows to add additional information to an existing process model based on data which is stored within the log. Areas of application are the adaptation of, for example, execution models like the model from a WFMS or to add information to models which are used for simulation purposes. For example if the log contains timestamps it is possible to add this information to the model and with simulation new information like cycle time or bottlenecks can be discovered.

The difference analysis cannot be assigned to all of the above mentioned areas of application. First process discovery is used to extract the process model of two input processes. Then conformance is used to determine the differences between these two models with the adaptation that the difference is not expressed by quality criteria. In fact, it is represented through a generated graph. More details about the underlying concept gives the following section.

### 2.4 Conclusion

This section covers a discussion why a simple log file with only activities and traces stores enough information to build a difference graph. For the prototype implementation it is not possible to implement every available mining algorithm. Therefore a mining algorithm has to be chosen. The discussion showed that it is not adequate to use a specific mining technique to build a process model, because every one of them has advantages and disadvantages. Therefore the used categories to group the algorithms come in handy. With these three categories it is possible to select at least one algorithm from each category to have a wide spread of algorithms available for the prototype implementation.

## 3 Difference Graph Concept

Due to a variety of models that already exist within a company comparison of these models offers additional information. Comparing models based only with true or false is not suitable according to [53]. They propose a system that the similarity of models should be expressed by a number between 0 and 1. However, this method is not suitable to exactly show where the models are different or where they are equal. [43] targets to visualize these differences by representing the difference of two models using a new generated model called **difference model**. This approach compares all edges and nodes and creates a marking which states that if a node or edge is different or equal in the two models. To express the knowledge stored within the model it is visualized with specific visual properties. This concept will be the basis of the evaluation and implementation which is the main focus of the rest of this work.

### 3.1 Difference Model

The difference analysis can be seen as a subtraction of two process models. The result of this subtraction is a new model which was introduced by [43] and called difference model. It has similar properties to a process model, i.e., it has the same elements such as nodes and edges but both carry on additional entry named marking. This marking indicates the change of a node or an edge from one process model to the other. It is possible to distinguish between three or five markings.

**Three markings** (New, Unchanged, Deleted) are used if the process model does not contain weights on edges or nodes. An example of this is a Petri Net. The following list describes in which case which of the three markings is used.

- **New**, a node or edge is new if the node or edge does not exist in the first process model and exists in the second process model.
- **Unchanged**, a node or edge is unchanged if it exists in both input process models.
- **Deleted**, a node or edge is deleted if the node or edge appears in the first process model and does not exist in the second process model.

Figure 2 shows an example where the difference model is calculated.The two input process models are Input Model 1 and Input Model 2. In the example Input Model 1 is subtracted from Input Model 2. Input Model 1 shows node B which is not visible in Input Model 2. Therefore node B is marked with deleted in the difference model. On the other side Input Model 1 does not visualize the nodes C and D which are visible in Input Model 2. They get the marking New in the difference model. Nodes A and E are visible in both input models. Therefore the marking Unchanged is assigned to these nodes in the difference model. Besides markings on nodes Figure, 2 also visualizes the markings New and Deleted on edges. An example for an edge with marking Unchanged would be an edge between node A and E which is visible in both input models.

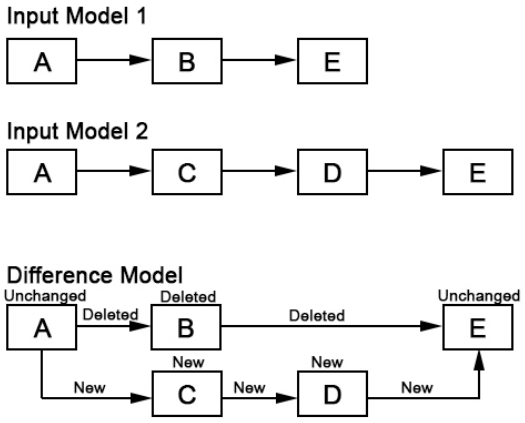



Figure 2: Difference calculation based on two input models without weights. The resulting difference model contains three different markings. Each marking is displayed above the activities and edges.

**Five markings** are used when the input process models contains weights. The two additional markings are (Positive changed, Negative changed). These two additional markings are used because the execution frequency of nodes and edges can be different. An example for weights used within a process model is a Heuristics Net. Here the weights represent the execution frequency. The following list describes the three markings from the difference model built without weights and the two additional markings.

– New, a node or edge is new if the node or edge does not exist in the first process model and exists in the second process model.

– **Positively changed**, a node or edge is positively changed if in the first process model it has a weight and in second process model the weight is higher.
– Unchanged, a node or edge is unchanged if it has the same weight in both input models.
– **Negatively changed**, a node or edge is negatively changed if in the first process model it has a weight and in second process model the weight is lower.
– Delted, a node or edge is deleted if the node or edge appears in the first process model and does not exist in the second process model.

Figure 3 shows an example where the difference model from two process models with weights is calculated. Markings Unchanged, Deleted and New are applied in the same way as mentioned before in the three markings section. The only difference is that using weights it is also possible to show weights for the markings Deleted and New in the difference model. Node B shows an Negatively Changed marking because the weight was reduced from Input Model1 to Input Model 2. Node C and all edges connecting node C are Positively changed because there weights has increased from Input Model 1 to Input Model 2.

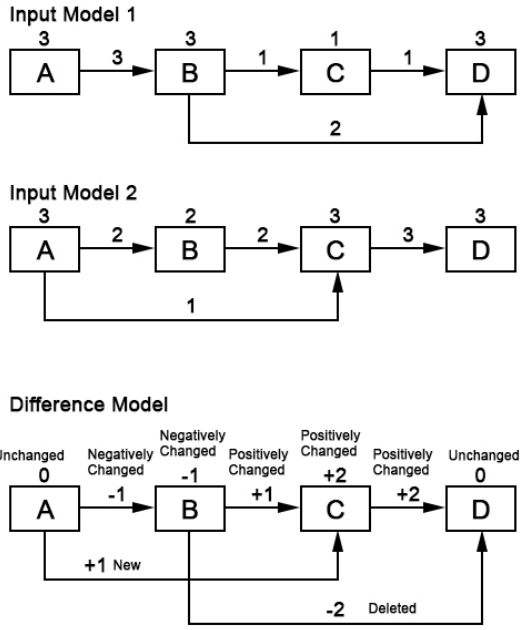



Figure 3: Difference calculation from two input models with weights. The resulting difference model consists of five different markings.

In this regard it is important to notice that each process model with weights can be transformed into a process model without weights. Whereas a process model without weights can not be transformed into a process model with weights. Hence it is possible to calculate both difference models, i.e., difference model with 3 and 5 markings for Heuristics Net. Figure 4 shows an example where the weights from Figure 3 are removed. The interesting thing about this is, that every edge and node which had a marking Positive or Negative changed now has the marking Unchanged. The markings New, Deleted and Unchanged from the initial five markings stay the same.

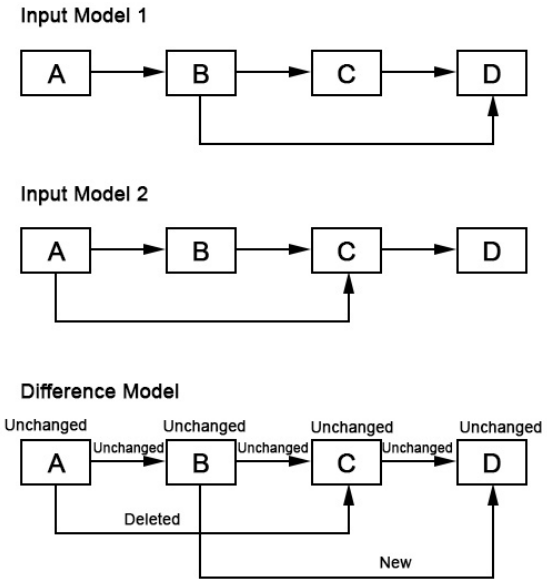



Figure 4: This figure is a transformed version of Figure 3. The weights where removed to show the differences between a process model with and without weights. Compared to the five applied markings from Figure 3 this version is only capable of representing three markings.

### 3.2 Instance Traffic

Each time a process is executed a so called process instance is created and if every step of the process is logged the traffic which is caused through the process is called instance traffic. A log file as shown in Table 1 represents this instance traffic from which it is possible to generate process models with mining algorithms. Every time a weight is visualized on nodes or edges it is the visualization of the instance traffic. The usage of the instance traffic has a positive effect on the difference analysis. [53] has identified that comparing two models generated

from instance traffic has an advantage over other process discovery methods. This approach leads to a unique result because the models visualize exactly the corresponding real world processes and give details about how the weights are distributed.

# 4 Difference Graph Visualization

Up to this point basics about different research areas where summarized and the difference graph concept was introduced. From now on the thesis addresses new research in terms of the difference graph which manly focuses on the visualization of the difference graph and the implementation of a corresponding prototype. At first a discussion should give details why the difference graph is visualized 2D and not 3D. Afterwards a literature research targeting different representation forms gives an overview which possible ways exist to visualize the difference graph. The section concludes with an example which covers every aspect, starting from the log file and closing with a color coded difference graph.

## 4.1 2D and 3D Comparison

Visualizing processes is not a new topic. Widely accepted and used are 2D directed graphs with different node and edge styles [44, 50, 54]. They often offer some kind of interaction like the fuzzy mining technique [12]. It simplifies the content visualized to the user. With given variables the user can determine how many information should be visible. The algorithm then groups the tasks on basis of different metrics. Another possibility to simplify process models is introduced by [40]. They use different visualizations which can be applied to different perspectives to support modeler and users as well as eliminate misunderstandings between these two.

Besides 2D visualization, 3D visualization is a rather uncommon type of visualization. The research area is divided. It is not clear if 2D visualization is better than 3D or vice versa. There are papers which show advantages of 3D. For example [38] state that 2D representations are limited in visualizing understandable information. Others such as [37] are a step ahead and show how collaborative work is possible within a 3D virtual world. On the other side, [6] state that 3D visualizations are hard to interact with and need a high-performance computer to render large processes. A statement from [25] is that 2D and 3D visualizations are not that easy comparable. The authors state that based on the skill level of the user large differences between 2D and 3D understandability emerge. In their survey they declare professionals with good knowledge about graphical user interfaces and novices with no professional computer experience and came to the result that professionals where able to interact and understand 3D graphical representations better then 2D graphical representations, novices on the other side had problems with 3D representations. There is a big gap in understanding a 3D representation between novices and professionals while a

2D representation is understood by professionals only slightly better than by novices.

Overall, a 2D representation was preferred because it offers several advantages. The primary is that 2D representations can be understood and interacted with in an easy way. This is very important because visualizing the difference adds new knowledge to an existing graph and if the user is not able to interact or understand the simple representation without the difference he will be overwhelmed with the difference graph. Usually adding additional information is an advantage of 3D graphs because they can visualize more information within one representation. However, the difference graph adds change information expressed by three or five markings and this can also be well visualized by 2D graphs. Some examples how this additional information can be visualized are presented by [41]. Examples are colors and edge width. The easier handling of 2D visualizations allows to have different views on the same graph or to visualize more than one graph, for example, the two input models and the output model.

Another point which should not be underestimated is the familiarity of the user with the 2D representations. The difference graph adds a new level of information to existing representations. These representations are calculated by different algorithms, for example, the fuzzy mining algorithm and are developed for 2D visualizations. Without changing the algorithms they will not make use of the additional space a 3D representation offers. Therefore to be able to generate 3D representations, each of the mining algorithms used for producing the difference graph has to be changed in order to support 3D representations. This familiarity is no coincidence as there is currently hardly any framework that allows a 3D representation. For example, [37] used the tool second life to create their collaborative modeling environment. If there is no uniform approach for 3D visualizations, representing change which goes a step further is not advisable and therefore a 2D representation is considered for the implementation provided in this thesis.

### 4.2 Research Approach

First, a literature research using the snowball method (track references of references) was performed. The paper [43] was used as a starting point. With the linked literature more related work how to visualize differences between two graphs was found. The list is available in Appendix A. This base helped to identify keywords (Table 2) which where used in two search engines, google schoolar [15] and microsoft academic [26]. The search engines lead to more relevant literature with new references. The full list of books, papers and websites can be found in Appendix B. From the found literature different representation forms how change in a graph can be visualized were extracted. Since the research has not answered which representation suits the visualization of a difference graph best an evaluation which is shown in Section 6 was conducted. Every found visualization approach is part of the evaluation since it was not possible to determine which is relevant and which is not. After the evaluation one representation will be

selected. The selected representation will then be implemented in the prototype implementation.

| | | |
|---|---|---|
| Graph differences | Calculate graphical differences | Matrix difference calculation |
| Delta Analysis | Union of Models | Difference computation |
| Visual Difference computation | UML diagram differences | Merging business processes |
| Merging processes | Structural differences between graphs | Visual graph comparison |
| Visualizing Differences between Graphs | Change detection | Change visualization |
| Graph evolution visualization | Visualization properties | |

Table 2: List of keywords used for literature research.

### 4.3 Overview on Representation Forms for Visualizing Difference

Important for visualizing the difference model are the markings. It is necessary that each marking has its individual style to be able to see the different meaning at a glance. This will help to quickly gain an overview of the model. A literature research showed many possible ways to visualize change information. The following list gives details about the extracted representation forms which are visualized in Figure 5.

- Edge and Node grayscale - colorizes the edge and node frame in five different colors from light gray to black.
- Edge width - expresses the five change markings with different width of the edge.
- Font size - every marking gets its own font size for node and edge label.
- Edge style - different styles from dotted to dashed are applied for every marking.
- Edge head style - the head of the arrow can have different styles like an filled arrow or an bent arrow.
- Node style - different shapes express every one of the five markings.
- Edge and Node symbols - colored symbols are visualized on each edge and node to express their marking.
- Font background color - every label on edges and nodes gets an individual background color.
- Edge, Node and Font color - color coding is used to express each marking with an individual color.

It is assumed that the user does prefer one of this representations over the others. Therefore an evaluation which is covered in Section 6 will give details

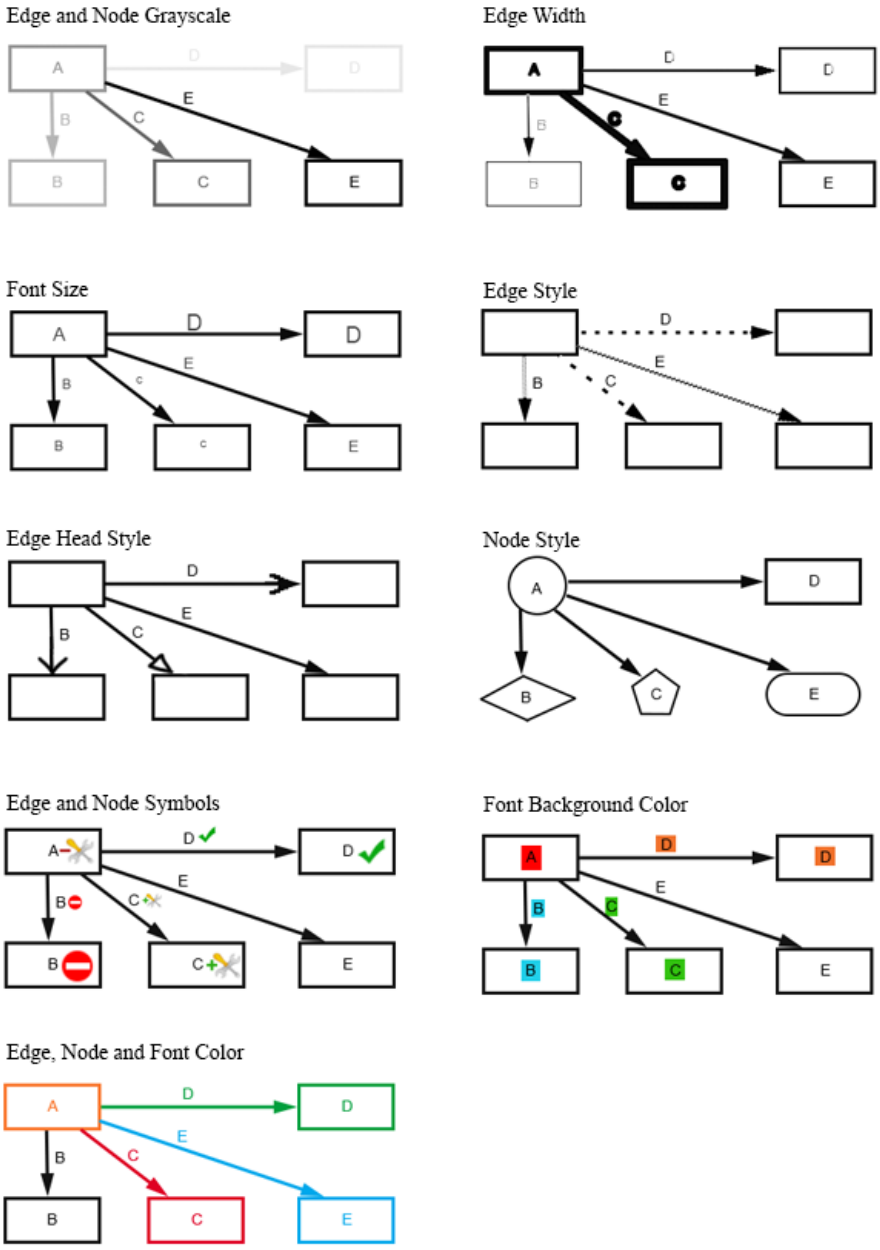



Figure 5: Extracted from the literature research, this figure shows nine ways how the difference graph can be visualized.

which representation is preferred and should be implemented in the prototype implementation.

### 4.4 Illustration of Concepts

To get a sens how all steps work together this section covers an example. Every aspect from the log file towards to the visualization of the difference graph is shown.

First of all, two process logs are needed. They are shown in Table 3. The left log is called „LogLeft“ and the right log is called „LogRight“. They consist of an equal amount of three traces and five different activities. As shown in „LogRight“ the traces do not have to be numbered. They can consist of any value. It is important that each process instance has its own trace identifier. Trace 1 of „LogLeft“ shows that activity „B“ is executed which is not executed in „LogRight“ and the only execution of activity „E“ is in „LogRight“. The overall start node stays the same but „LogRight“ has two different end nodes.

| LogLeft | |
|---|---|
| **Trace** | **Activity** |
| 1 | A |
| 1 | B |
| 1 | D |
| 2 | A |
| 2 | C |
| 3 | A |
| 2 | D |
| 3 | C |
| 3 | D |

| LogRight | |
|---|---|
| **Trace** | **Activity** |
| Alpha | A |
| Beta | A |
| Alpha | C |
| Gamma | A |
| Beta | C |
| Alpha | E |
| Beta | D |
| Gamma | C |
| Gamma | E |

Table 3: Two log files with nine entries and overall five different activities. The log on the right shows that the identifier of a trace does not have to be a number.

For the difference analysis two process models are needed, therefore two log files are mined. In this example the heuristics miner is used which generates the two graphical representations for the above shown log files. On the left of Figure 6 is the corresponding representation of „LogLeft“ and on the right for „LogRight“. The image shows that both mined graphical representations start with the same activity „A“ and an instance traffic of three, then the left representation splits the traffic, one instance executes activity „B“ while the other two execute activity „C“. On the last step „B“ and „C“ lead to activity „D“ which has an traffic of three. On the other representation a traffic of three goes from „A“ to „C“ and then splits to the two end activities „D“ with a traffic of one and „E“ with a traffic of two. It was mentioned before that „LogRight“ consists of two end Nodes, this

is not easy to see from the log file and even not if the log is much larger, the two end nodes can be identified pretty easy in the graphical representation. One thing to notice is that if a node is not a start or end node the total weight of incoming edges, outgoing edges and weight of the node have to be equal.

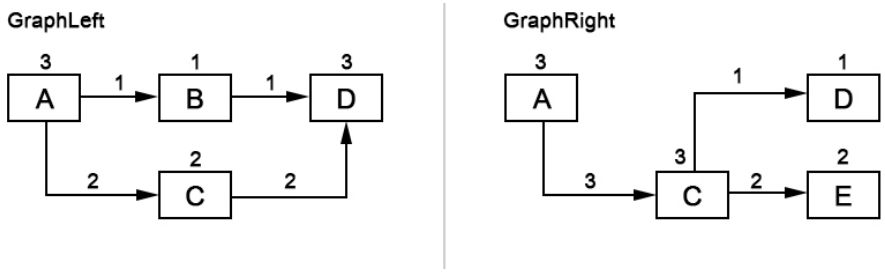



Figure 6: Mined representation of the log files shown in Table 3. Each representation consists of four activities and weights on nodes and edges.

With these graphical representations the difference model can be calculated. Therefore the instance traffic of a node from the left representation is subtracted from its corresponding node of the right representation. If a node is not available in one of the representations its traffic is set to 0. Same goes with edges because edges do not have a label. They are identified by their start and end activity. Table 4 shows the calculations for every node and edge as well as the individual marking.

Here is a short recap how the markings are applied, every time a node or edge from one graph is not present at the other it is either new or deleted. New if the node is present in the right representation otherwise it is deleted. If the node or edge weights match exactly it is unchanged and if it does not match it is positively or negatively changed. A positively change is acquired if the weight of the left representation has a lower value. If the weight of the right representation is higher then the marking negatively changed is applied.

These change operations can be transformed into a graphical representation. Figure 7 shows a color coded version for change visualization. Every color represents a specific marking, the model consists of five different colors because the input models had weights and therefore it was possible to distinguish between five markings. The figure shows that all nodes of the two input models are visualized. This is important even if node „B“ has the marking deleted. It is important to visualize this node because otherwise the difference graph will have a lack of information.

| Nodes | | | |
|---|---|---|---|
| Name | Values | Result | Marking |
| A | 3 - 3 | 0 | Unchanged |
| B | 0 - 1 | -1 | Deleted |
| C | 3 - 2 | 1 | Positively changed |
| D | 1 - 3 | -2 | Negatively changed |
| E | 2 - 0 | 2 | New |

| Edges | | | |
|---|---|---|---|
| Name | Values | Result | Marking |
| A to B | 0 - 1 | -1 | Deleted |
| A to C | 3 - 2 | 1 | Positively changed |
| B to D | 0 - 1 | -1 | Deleted |
| C to D | 1 -2 | -1 | Negatively changed |
| C to E | 2 - 0 | 2 | New |

Table 4: Difference calculation for nodes and edges from the mined process model shown in Figure 6. Name is the identifier, Values are the corresponding weights from the two models, Result represents the outcome of the calculation and Marking is the applied marking for this node or edge.

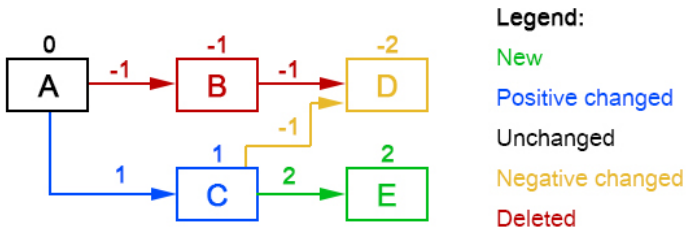



Figure 7: Color coded representation of the calculated difference model presented in Table 4. A legend is shown on the right side to be able to distinguish between the five markings which are represented through colors.

This section gave an overview how the difference model is calculated and how the difference graph could be visualized. The next sections will cover some requirements to be able to generate a difference model and improvements to enhance understandability and useability of the visualization.

## 5 Difference Model Requirements and Recommendations

In order to support the implementation process some requirements for the difference model and recommendations for the visualization will be defined. The recommendations address useability and understandability of the difference graph. This is achieved by bringing a uniform structure into the representation. This chapter contributes the main conceptual contribution of the thesis.

### 5.1 Model Requirements

As mentioned before, process model and difference model consist of nodes and edges. Each node has to be labeled with a unique name. If the name is not unique then a unique id has to be created and stored. All nodes have to be on a path between start and end node where the start node is the only node which has no incoming edge while the end node is the only one which has no outgoing edge. Each edge connects one node with a second and can be seen as tuple (Node, Node). [43]

Compared to the above mentioned definition, there are some changes and additional requirements for difference analysis. At least two process models are required to be able to calculate the difference model. This thesis only focuses on difference analysis between two models. For three or more there is still research to do. It is possible to obtain two process models from a single process, for example, if one model is representing year 2013 and the other model 2014 (versioning). To build a meaningful difference graph it is important that there is at least one node which matches between the two process models. If no node matches then its not possible to calculate a difference. The difference graph would visualize both models without any calculation. This would lead to an output which is the same as the input.

There are two reasons why the requirement for one defined start and one defined end node has to be extended for the difference model. The first is that processes mined from logs may just have different start and end points [55]. For example two workers get the task to produce a desk. The first starts with producing the four table legs and then he produces the tabletop. The other one starts with the tabletop and ends with the four table legs.

In an other case the difference of two process models with one defined start and end point is calculated, but leads to a difference model with two start and end nodes. It might even be possible that the difference model does not contain an end or a start node anymore. Figure 8 shows an example. The difference graph does not contain a node without outgoing edges, hence no end node.

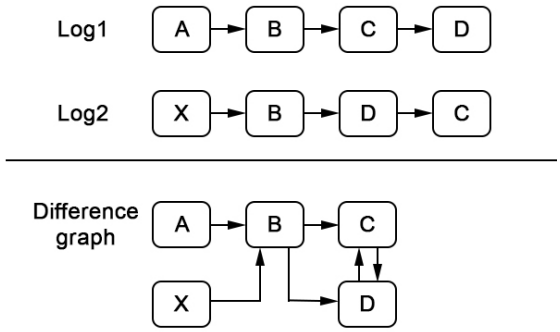



Figure 8: The difference graph is calculated from two input models and consists of two start nodes and no end node. One thing to mention here is that node „X“ and edges from „X“ to „B“, „B“ to „D“ and „D“ to „C“ are marked as deleted.

### 5.2 Design Choices

The following design choices show how to improve useability and understandability of the difference graph. Figure 9 shows an evolution scenario where the three steps Graph positioning, Node visualization and Positioning of nodes are applied. All other recommendations (Change visualization, Complete output and Legend) are shown in every of the three examples.

**Visualization of Input and Graph Positioning** Visualizing only the difference graph is not enough, as the user cannot understand why the change operations are visualized in a specific way. To secure a clear understanding it is important to visualize the input from which the difference was calculated. The user then can trace back how the difference graph was calculated and gets a clear understanding of the difference graph. For example, Figure 7 on page 19 shows a difference graph without displaying the original models. Hence, it is difficult to understand why node „D“ has the marking „Negatively changed“ with a weight of -2.

Visualizing the two original models and the difference graph leads to three models which have to be displayed. There are different attempts how they can be positioned. For example, all three side by side, one below the other, two on top one on bottom, two left one right. [21] use in their merge scenario the attempt to position the input on top and the merged model on bottom. This approach seems to be a good practice in terms of graph merging and will be used for the difference graph. None of the inputs is more important than the other. The important thing is to focus the users attention on the difference graph. Therefore each of the inputs gets 20% of the space available on the screen. The remaining 60% are reserved for the difference graph. As mentioned before the two inputs

are positioned side by side and the difference graph on bottom of them. By putting the input side by side this representation should also allow an easy back tracking how the difference is calculated.

**Node Visualizations** Visualizing nodes in the difference graph offers two possible ways.

One is that each graph visualizes nodes with its own style this can result in three different node visualizations. Figure 9 „Graph positioning“ shows an example where every node from „Input1“ is visualized as rectangle, „Input2“ as circle and the „Differencegraph“ nodes as rounded rectangle.

The second way is that each graph visualizes nodes in the same way. Figure 9 „Node visualizations“ shows that each node from „Input1“, „Input2“ and „Differencegraph“ are visualized as rounded rectangle.

Using the same style for every graph should support that the user is able to understand the difference graph faster and does not have to think about why there are different styles for nodes, edges and fonts. It´s also one way to support the mental map [22]. The implementation covered by this thesis will style every node in the same way. Attention should be paid that this is only possible if the different styles do not express different behavior.

**Positioning of nodes** There are two ways how the nodes can be positioned within the graphical representations.

One way is using a positioning algorithm for each graph. Positioning algorithms like force-directed placement [48], Genetic algorithms [46] and others [49] allow, for example, efficient use of space, no overlapping edges, no overlapping nodes. Using a positioning algorithm for each graph leads to individual representations. Figure 9 „Node visualization“ shows an example where the positioning of the nodes is individual in each representation. For example node „C“ is positioned on top right in „Input1“ and „Input2“. In the „Differencegraph“ node „C“ is positioned on the bottom center.

Another way is to use the same position for same nodes in every representation. Figure 9 „Positioning of nodes“ shows that nodes which appear in two or more representations are in each of these representations on the same position. Positioning every node on a unique place over all three graphs can also lead to no overlapping nodes and edges as well as other advantages mentioned in the individual positioning paragraph.

[16] concluded the mental map is a factor which should not be underestimated in terms of graphs. Using the same position for the same node will support the mental map and increase the understanding of the difference graph. When every node is on the same position over all three representations, backtracking how the difference graph is generated is easier compared to different positions.

**Change visualization** As mentioned before it is crucial that every change operation (New, Positively changed, Unchanged, Negatively changed and Deleted) gets its own style. An individual style for every change operation helps to

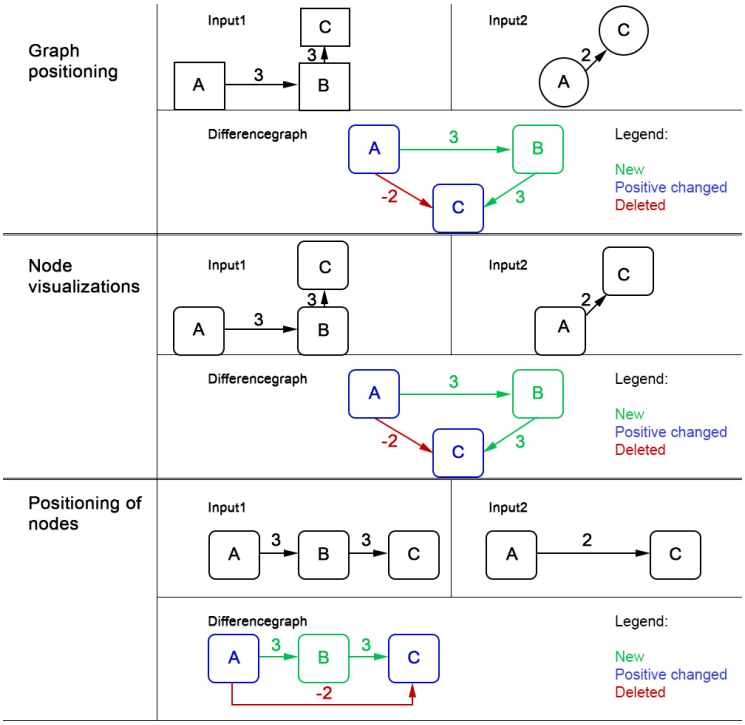



Figure 9: Design choices how to improve understandability of the difference graph. The image shows an evolution where the keyword from the left is applied to the representation on the right.

distinguish between the different change operations. To secure a consistent and intuitive understanding Section 6 covers an evaluation to find the best fitting representation for change operations.

**Complete output** This choice is especially targeting the difference graph it is important that the difference graph visualizes all edges and nodes even if they where deleted or have not changed by calculating the difference.

For example Figure 9 „Input2“ shows an edge from „A“ to „C“ „Input1“ does not show the edge this gives two possibilities first the edge could be marked as new or the edge could be marked as deleted. In this example the second case is assumed and when the edge war deleted why should it be visualized? When handling the difference graph it can be important to exactly know that this edge was deleted from process two to process one and if it is not visualized in the graph how should the user know that it was deleted? Therefore all nodes and edges from the inputs have to be visualized in the difference graph.

**Legend** Not only is it important to show the user that there was a change and visualize it in different styles. It is also important to show the user what this style stands for. Therefore the legend should cover the weight dependent three or five markings which can be visualized by the difference graph.

## 6 Evaluation

Besides the concept which was discussed in Section 3 on page 9 the visualization is an important part of the difference graph. The literature research shown in Section 4.2 on page 14 tried to answer which visualization should be used for the difference graph. However, no visualization was tailored to the difference graph. For this reason different visualizations where extracted from the literature which are shown in Figure 5 on page 5. For the difference graph it is necessary to find a representation which meets the expectations of the user. To meet the expectations as good as possible a survey was conducted.

Goal of this survey is to find the best suiting representation for the difference graph. Besides the representation it is important to know if the concept of the difference graph can be understood.

To find the best suiting representation several research questions have to be answered:

- Is the difference graph understandable?
- Which representation should be used for visualizing the difference between two models?
- What should be visualized by the difference graph?
- Is the difference graph a good choice for comparing process models?

### 6.1 Survey Tool

Searching for a tool which is capable of creating a survey leads to different tools LimeSurvey [8], SurveyMonkey [35], MaQ Online [47] and many more. To evaluate which tool should be used requirements are defined:

- Questions should allow to insert pictures. This is necessary because the different representation styles have to be visualized in order to ask the user.
- Answer types should be simple text, check box, matrix and some kind of ranking system. The matrix will be used on the individual pages for each representation and the ranking is needed as follow up to rank the representations after seeing them all.
- Exporting the answers in any format is required as long as it can be transformed into CSV. Beneficial is a direct export as CSV or SPS/SAV.
- The tool has to be free of charge.

| Tool | Pictures in Questions allowed | All Answer Types | Export | Export as CSV or SPS/SAV | Free of charge |
|---|---|---|---|---|---|
| MaQ Online | | | X | | X |
| SurveyMonkey | | X | X | X | |
| LimeSurvey | X | X | X | X | X |

Table 5: This table shows which requirements where meet by which tool.

Table 5 shows which requirements where meet by the tools. In comparison with the other tools Limesurvey met all of the above mentioned requirements. Limesurvey has the following advantages:

- Allows HTML code in questions.
- From the evaluated tools Limesurvey offers the most answer types.
- Extensive export and import formats.
- Editor with preview for each question.
- Possibility to add some logic, for example, when question A was answered in a specific way question B was visible.

### 6.2 Survey Design

After installing LimeSurvey on a web server the layout was defined. Key aspects are enough space between questions this should secure that the user does not feel cornered. A font-size which has no serifs and is clear to read secures that the user has no problem reading the questions or answers. Single pages for categories, for example, first the user gets an introduction and then a button opens the next page. Single pages have two reasons. First, the attendee does not see the whole

survey and is not frightened by the surveys length. Second, on each submit of a page the input can be checked for completeness and when something is missing the attendee finds the missing entries without a lot of scrolling.

**Survey Introduction** To secure that every attendee has a principal understanding what is meant by a difference graph. The first part of the survey is devoted to an example. The example illustrates two process models and the calculated difference graph.

After the example there is an question which checks if the user has understood the principle of the difference calculation. Another question asks for general graph knowledge. Both questions will be interesting to associate with data collected during the survey. It could be, for example, investigated whether user with higher knowledge of graphs prefer different styles then user with lower knowledge.

**Survey Representation Styles** Representation styles are the main part of the survey. Each representation from Section 4.3 on page 15 will be visualized to the user on a single page. Figure 10 shows the structure of one page. The first questions asks the attendee how expressive the visualization in terms of changes on nodes and edges is. To secure that the attendee has to think about the representation the second question shows a 5x5 matrix where the appropriate style for each node and edge has to be set.

Afterwards the attendee is asked if other styles (in the case of Figure 10 other symbols) would be better. If the answer is yes then an additional question opens where the style for each marking can be described. The last question on each page is open for comments.

Last part of this group is a ranking where each representation style is listed once. The user has to drag them into the order from best to worst. This ranking is very important because at this point of the survey the user has seen every representation once.

**Survey Details and Demographic Questions** Questions about combining different representations and if the visualization should change based on the complexity of the graph are asked in the details section. On the next page a question asks for process knowledge. Depending on the answer to the question about process knowledge a second question is shown. The second question asks if the difference graph can be useful in the context of process models. The survey is closing with some demographic questions on the last page.

### 6.3 Procedure

Each question from the survey was developed and checked with the „Ten Commandments“ from, by, Porst [36]. The „Ten Commandments“ give advice on wording, complexity of questions, dos and don‘ts. In order to review the survey

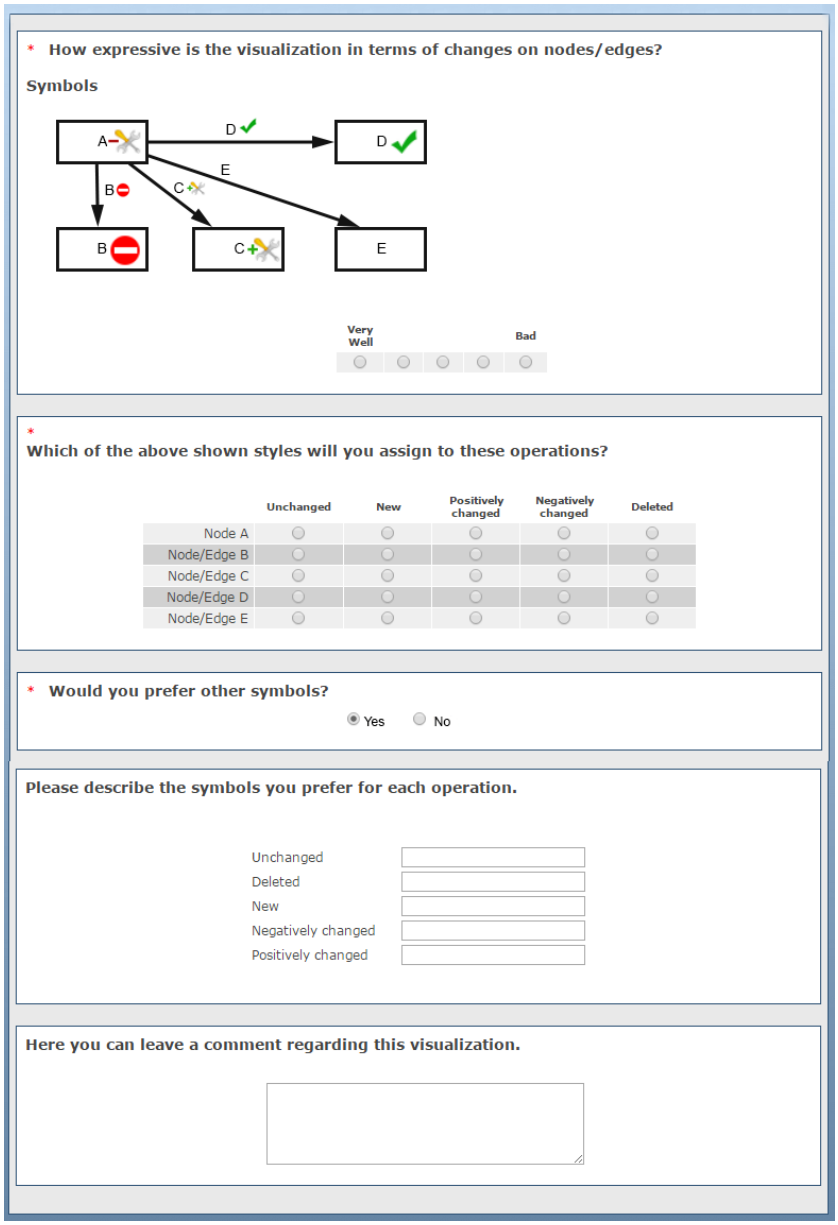



Figure 10: This Figure shows an page from the survey. For better understanding the text was translated from German to English.

before the URL is given to the attendees a pretest was performed. [42] present how their pretest worked and that it had an overall positive effect on the survey. To check if the questions are understood correctly, the questions from this survey where part of a two-round pretest.

**First Pretest** After styling and categorizing the questions into groups a discussion with two participants took place. Every question and answer was analyzed. Some of them where slightly changed to better fit into the survey. The first representation shown to the user has undergone the most changes. The questions where changed and a second page was added. On the second page an example representation is given to help securing the understanding what the surveys goal is.

**Second Pretest** The second phase took place with five participants. None of them was involved in the first pretest or in any kind of the surveys creation process or difference graph development. While they filled out the survey they where observed. Their task was to always think aloud and if something was not clear to ask questions. These questions where not answered they where only noted. From the five participants two had not worked with graphs at all, one had worked with graphs but not in context of processes and two had worked with graphs and processes. After completing the survey some questions where asked to improve the quality of the survey.

This pretest showed that four of five users where irritated by the task to declare which node represents which marking and which edge represents which marking because it was always the same. The observation showed that on the first pages the attendee looked at the graph and answered the questions for nodes and edges, after the third to fourth page every one just copied the answer from the node question to the edge question and has not looked at the graph again. Therefore these two questions where merged into one question and the details group was extended with a question if nodes and edges should have the same style. This also helps solving another issue. The overall feeling was that the survey is too long, the average time was 37 minutes. Another solution would have been to delete specific non relevant representations but only two of the five attendees said that they would delete a representation. They did not mention which representation they would delete. Therefore no representation was deleted.

### 6.4 Sample

The target audience is set between 20 and 50 years and can be elementary students, students as well as worker. From the average worker it is expected to perform the survey unbiased. This group is normally not familiar with the topic of graph analysis and business processes. It is expected to get another point of view from this group then from students. Students normally have worked with graphs, for example, in mathematics and business related areas of education. From them it is expected to understand the concept of the difference graph

faster then the average worker. An interesting observation will be to see if the answers of the two groups diverge or not.

To target this audience the URL to the survey was manly distributed in companies and universities. On the first day the survey went live the URL was sent to the following companies, an insurance company (20 attendees), a software developer (60 attendees), a police office (40 attendees) and a lawyer (12 attendees). On the second day an email was sent to 27 business information technology students. After one month the URL was posted in a group on Facebook addressing students. While the survey was running students at the university where asked if they would complete the survey.

The survey started on 5 June 2014 and ended on 18 August 2014. Overall, 73 persons started the survey which means they have at least passed the introduction page and 31 of them completed the entire survey. From the 42 which have started but not finished the survey only 12 have entered any value by any question. For the further evaluation the 31 persons which have completed the survey are considered.

**Target Audience** This evaluation shows if the target audience was meet. The questions where optional and it is possible that a person did not answer the question. From the 31 attendees 27 (87 %) are in the range of 20 to 50 years, one (3%) was 19 years and 3 (10%) did not answer the question. The next question addresses the employment status of the attendee, 1 (3%) Person has not answered the question, 8 (26%) where without an employment and 22 (71%) where with employment. A true (18 (58%) attendees), false (12 (39%) attendees), no answer (1 (3%) attendee) question should give details if the attendee is a student or not. Then a cross tabulation between students and employe showed that each person which is not employed is at least a student.

**Time** The pretests have shown that for most participants the time was too high. Therefore some changes from the pretest to the final survey where made. This evaluation will show if the changes had an positive or negative affect on time. Figure 11 shows a box plot which visualizes the time with a 50% percentile of 25 minutes an 7 seconds. The lower percentile 25% is 20 minutes 42 seconds and the upper percentile 75% 36 minutes 46 seconds. The mean-time from the pretests was 37 minutes. The mean-time from the final survey was 28 minutes. Compared to the pretest this is an reduction of about 25 percent. Overall the changes from the pretest to the final survey had a positive impact on the mean-time.

### 6.5 Results

The answers were exported from LimeSurvey and imported to SPSS. Each answer group was categorized as nominal or ordinal scales to allow a statistical correct evaluation. The feedback given from open questions was exported and will be handled separately. The questions from the survey will be assigned to the corresponding research questions.

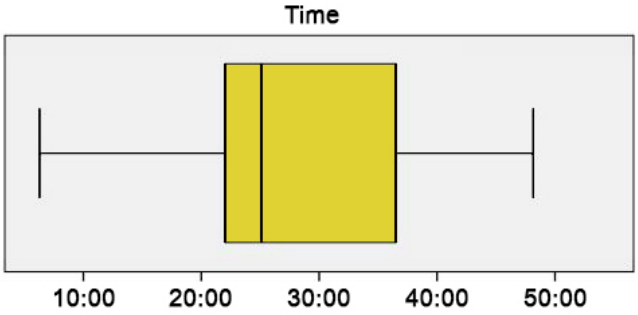



Figure 11: Boxplot which visualizes the time for completing the survey.

**Is the difference graph understandable?** To answer this question the survey showed an short example how the difference graph is calculated. Directly after the example a question checks if the user has understood the example. The question asks the user why a specific edge has a given weight and proposes five different answers where one is true and the others are false.

The pie chart from Figure 12 shows that 19 attendees (61.3%)(orange) of the answers where correct and 12 attendees (39.7%)(red, purple and blue) decided for the wrong answer and have not understood the example. No attendee decided for answer three which was also wrong.

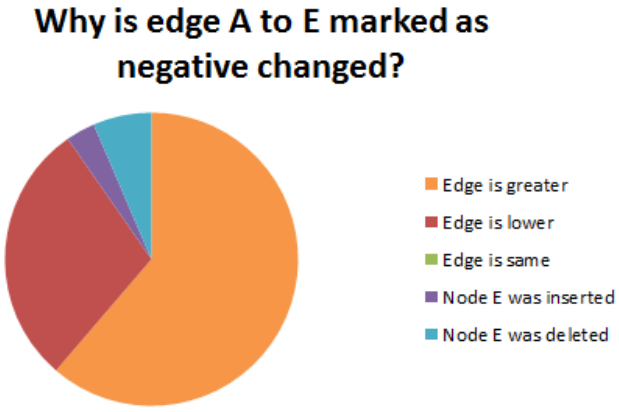



Figure 12: A pie chart which visualizes the general understanding of the difference graph example, orange was the right answer and red was nearly right.

An interesting combination is to compare the knowledge the attendees have, with the percentage of correctly / not correctly answered questions. The result shows that 14 of 19 (73.6%) attendees which worked with graphs before answered the question right. Against 5 of 12 (41.6%) correct answers from attendees which have not worked with graphs before. The Perason Chi-Square test with an significance value of 7.5% shows that there is only a small correlation between the right answer and the graph knowledge.

**Which representation should be used for visualizing the difference between two models?** The survey includes different questions do target this research question. The first approach to answer this is by a **rating**. Afterwards a matrix for **intuitive understanding** and a **ranking** are evaluated.

**Rating** Each visualization from Section 4.3 on page 15 offers a rating from good to bad. To be able to compare these ratings they where transformed into numbers from 5 (good) to 1 (bad) then the mean-value was calculated. Figure 13 shows that three representations are on top and only a difference of 2.3 percent separates „Symbols“ with a rating of 4.23 and „Edge Node Font Color“ with a rating of 4.13.

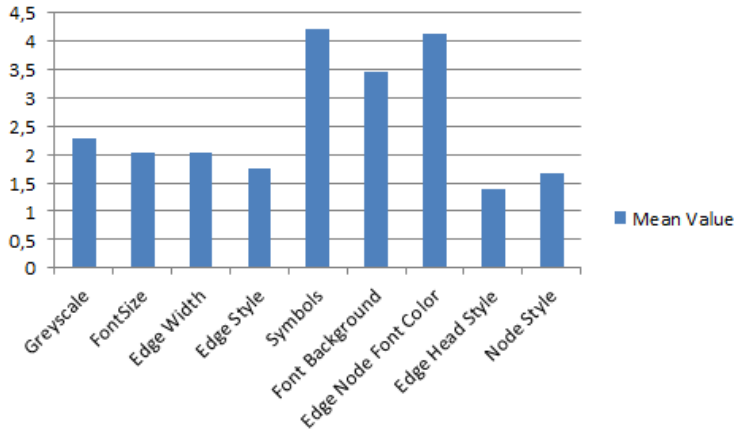



Figure 13: Mean value of ratings, high values are good low values are bad.

To analyze if there are coherencies between two variables an analysis with the correlation coefficient by Pearson was made. The result shows a statistical link between „Edge Node Font Color“ and „Font Background Color“. This link is significant at the 1% level. The correlation is positive with a value of 0.799. This means the higher „Edge Node Font Color“ is rated, the higher „Font Background Color“ is rated. This will be relevant by selecting the final representation. If

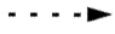
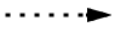
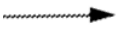
similar results also show this behavior then there is no reason to chose „Font Background Color“ over „Edge Node Font Color“.

Another interesting combination is to see if attendees with graph knowledge have rated differently from attendees without graph knowledge. To answer this a chi square test was performed. In some cases like „Font Size“ (significance level 5%) „Edge Style“ (significance level 5%) and „Edge Head Style“ (significance level 1%) the answer is yes. There is a relation between the rating of the user and the graph knowledge.

**Intuitive Understanding** Besides an overall rating the user had to determine which node and edge pair represent which marking. This allows to find an intuitive representation and also to build an appropriate legend. For example Section 4.4 on page 18 shows a color coded representation. The attendee was given a similar representation without a legend. Then the attendee had to answer which marking is applied to which node and edge. To answer this a matrix which consists of columns for the nodes/edges and rows for markings was given. Each row was allowed to have one entry.

For each visualization every representation style is displayed in the second column of Table 6. The following columns are the five markings. The sum for each representation style and marking was build and can be seen in the table. For example the first representation style Node A from the „Greyscale“ visualization had 22 attendees which interpreted this style as Unchanged.

| Name | Representation style | New | Positively changed | Unchanged | Negatively changed | Deleted |
|---|---|---|---|---|---|---|
| Greyscale |  | 1 | 2 | 22 | 3 | 3 |
| |  | 2 | 2 | 2 | 20 | 5 |
| |  | 6 | 20 | 3 | 0 | 2 |
| |  | 3 | 1 | 3 | 5 | 19 |
| |  | 21 | 5 | 2 | 0 | 3 |
| FontSize |  | 5 | 10 | 15 | 0 | 1 |
| |  | 1 | 2 | 3 | 22 | 3 |
| |  | 0 | 2 | 5 | 4 | 20 |
| |  | 17 | 5 | 6 | 2 | 1 |
| |  | 5 | 9 | 13 | 2 | 2 |
| |  | 8 | 17 | 6 | 0 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| EdgeWidth |  | 2 | 1 | 3 | 8 | 17 |
| |  | 20 | 6 | 2 | 2 | 1 |
| |  | 0 | 5 | 6 | 13 | 7 |
| |  | 2 | 4 | 17 | 6 | 2 |
| EdgeStyle |  | 7 | 4 | 6 | 9 | 5 |
| |  | 6 | 7 | 2 | 10 | 6 |
| |  | 8 | 8 | 4 | 7 | 4 |
| |  | 6 | 9 | 5 | 2 | 9 |
| Symbols |  | 0 | 1 | 1 | 27 | 2 |
| |  | 1 | 0 | 0 | 2 | 28 |
| |  | 1 | 27 | 1 | 1 | 1 |
| |  | 27 | 3 | 1 | 0 | 0 |
| |  | 1 | 0 | 29 | 1 | 0 |
| Font Background |  | 0 | 1 | 0 | 4 | 26 |
| |  | 10 | 21 | 0 | 0 | 0 |
| |  | 21 | 9 | 1 | 0 | 0 |
| |  | 0 | 2 | 1 | 24 | 4 |
| |  | 0 | 0 | 29 | 1 | 1 |
| Edge Node Font Color |  | 0 | 0 | 4 | 26 | 1 |
| |  | 3 | 0 | 26 | 1 | 1 |
| |  | 0 | 1 | 0 | 3 | 27 |
| |  | 23 | 7 | 0 | 1 | 0 |
| |  | 5 | 22 | 0 | 2 | 2 |

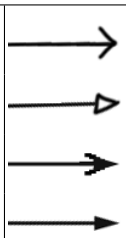
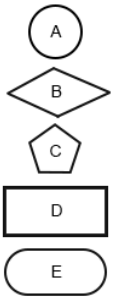
| | | | | | | |
|---|---|---|---|---|---|---|
| Edge Head Style |  | 8 | 8 | 9 | 2 | 4 |
| | | 8 | 5 | 2 | 10 | 6 |
| | | 8 | 12 | 5 | 4 | 2 |
| | | 4 | 2 | 16 | 8 | 1 |
| Node Style |   | 6 | 5 | 8 | 7 | 5 |
| | | 12 | 6 | 2 | 8 | 3 |
| | | 8 | 10 | 3 | 4 | 6 |
| | | 2 | 1 | 22 | 2 | 4 |
| | | 1 | 7 | 5 | 7 | 11 |

Table 6: This table shows for each marking which representation style represents it.

Table 6 shows every value for each style. To be able to compare the markings, the best four representation styles for each marking are shown in Table 7. In each category on top is the „Symbols“ visualization. A mean value of 27.6 attendees (89%) had the same intuitive understanding which marking is represented by the representation style. In the second row „Edge Node Font Color“ is dominating four of the five styles are from the „Edge Node Font Color“ representation style. With a mean value of 24.8 attendees (80%) „Edge Node Font Color“ inherits the second highest rating. Directly following the „Edge Node Font Color“ style is the „Font Background“ style with a mean value of 24.2 attendees (78%). The first three visualizations of the fourth row are from the „Greyscale“ style with an mean value of 20.4 attendees (65.8%). The other two visualizations are from the „Font Size“ style which inherits the fifth place with 17.4 attendees (52.2%). For the other representation styles the overall agreement is below 50%.

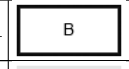
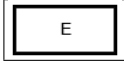
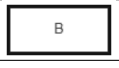
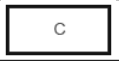
| New | | Positively changed | | Unchanged | | Negatively changed | | Deleted | |
|---|---|---|---|---|---|---|---|---|---|
|   | 27 |   | 27 |   | 29 |   | 27 |   | 28 |
|   | 23 |   | 22 |   | 29 |   | 26 |   | 27 |
|   | 21 |   | 21 |   | 26 |   | 24 |   | 26 |
|   | 21 |   | 20 |   | 22 |   | 22 |   | 20 |

Table 7: For each marking the top four representation styles are visualized.

**Ranking** After the individual visualizations the user had to rank all visualizations from best to worst. To be able to calculate which representations are on top and which on bottom the rankings were converted to numbers from one to nine where one stands for the worst and nine for the best this transformation allows to build the mean value for each visualization. On top of figure 14 are the representations „Edge Node Font Color“ with 7.80 points (86%) „Symbols“ with 7.48 points (83%) and „Font background color“ with 6.87 points (76.3%). This is approving the evaluations which where made before where these three visualizations always stayed on top. The difference is that in this evaluation „Edge Node Font Color“ is on top and not „Symbols“ which was on top in the two evaluations before. Interesting to note is that the ranking took place after the individual visualizations, this means the user has seen every representation once. This result messes a little with the results from before. The only thing to say clear is that for the prototype implementation one of these top three representations will be considered.

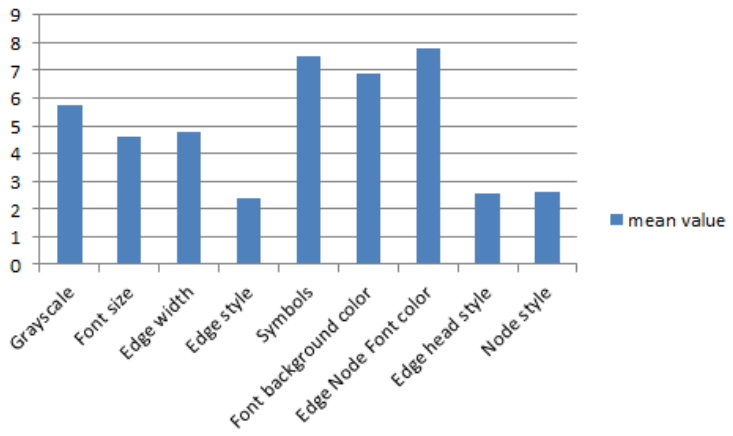



Figure 14: Bar chart which shows the transformed ranking of the visualizations.

As mentioned in the thirst evaluation of this section. There was a high correlation between „Edge Node Font Color“ and „Font Background Color“. It was assumed that the rating will also show this correlation. As before the Pearson Correlation coefficient was used. The result showed no correlation (significance level of 35%) between the ratings „Edge Node Font Color“ and „Font Background“

An interesting observation based on the ranking data shows Figure 15 where not only the ranking is considered. The ranking is merged with data of attendees which have worked with graphs and which not. The figure shows that the top four are generally higher rated by attendees which worked with graphs, for example, the highest rating with 8.36 points (93%) has „Edge Node Font Color“ this is an increase of 20 percent.

This observation does also show a cross tab between the ranking of „Edge Node Font Color“ and the graph knowledge. With an significance value of 5% the Pearson Chi-Square test shows that there exists an relation between those two variables. The ranking of „Grayscale“ showed nearly the same significance value while for all other representations no relation between the ranking and the graph knowledge was found.

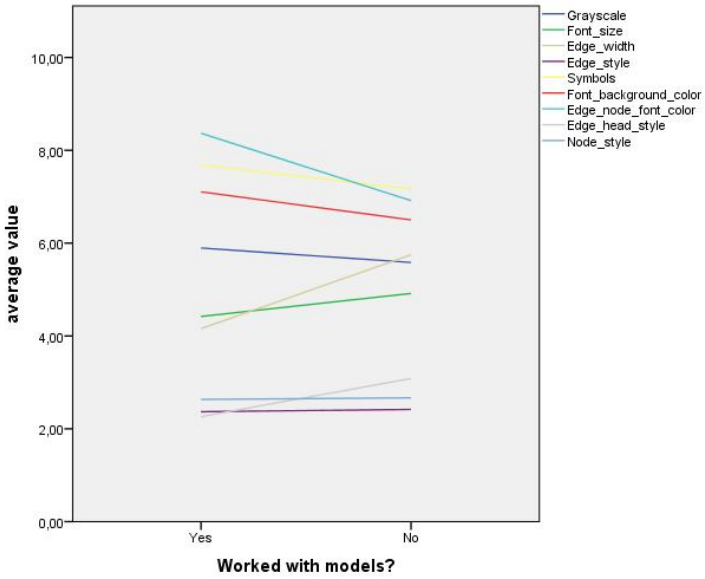



Figure 15: Evaluation which shows the rankings distinct by attendees which have worked with graphs and which have not.

**What should be visualized by the difference graph?** After the different representations the user was asked if positively and negatively change on edges and nodes should be visualized. 29 of 31 attendees (93.5%) answered positively and negatively change should be visualized. Another question at the end of the survey was if edges and nodes should be visualized with the same representation style. 24 (77.4%) attendees said yes while 7 (22.6%) of the attendees answered with no.

Maybe some representations are better suited for small graphs and others better for big graphs. There are 22 attendees (71%) which answered they would not differentiate between small and big graphs. Nine (29%) attendees think that there should be different representations for small and big graphs. Four attendees answered that small graphs should be visualized with „Symbols“, two said „Grayscale“ and „Edge Width“, „Edge Node Font Color“, „Edge Head Style“ where mentioned once. According to four attendees big graphs should be visualized by „Edge Node Font Color“, two mentioned they would prefer „Symbols“ and one person each said „Edge Style“, „Font Size“ and „Grayscale“. As before „Symbols“ and „Edge Node Font Color“ match for top places but in different categories.

**Is the difference graph a good choice for comparing process models?** One of the first questions of the survey asked the attendee if he/she had/has worked with graphs before, 19 of 31 attendees answered with yes and 12 said no. The second question was if the attendee has worked with processes 13 of 31 attendees have done this. The third one was only visualized to attendees which have answered the second question with yes therefore only 13 attendees where able to answer it. The question was if they think the difference graph can be meaningful applied on process models, 11 of 13 attendees answered with yes this is an agreement of 84.6 percent of the attendees who have experience with processes.

**Comments** Each visualization provided a field for further comments most of the comments where that the representation was either good or bad but there where some other comments which are summed up in the next paragraphs.

One comment for „Grayscale“ representation was that with different brightness settings and user vision it could get very hard to see that there are different colors.

For the „Edge Width“ representation two comments said that positive and negative change are hard to distinguish and therefore only new, deleted and unchanged should be visualized.

Two attendees mentioned that „Edge Node Font Color“ is much better then „Font Background Color“ the understanding is nearly the same because the colors are the same. „Edge Node Font Color“ is more memorable then only „Font Background Color“ which does not look very pleasant.

One comment mentioned that for the „Node Style“ representation more different styles would be better.

Two attendees gave a comment targeting the effective usage of a difference graph with processes. One was that it would be easier to analyze partially different processes. The other was that by building the difference it would be easy to see if the processes can be merged.

Two attendees suggested that the use of a legend would be really good to help understand different representations.

### 6.6 Discussion

The survey showed that visualizing change through a difference graph is a proper usage. Two examples from the comments how to use the difference graph are comparing partially different processes and getting an overview if two processes can be merged and where problems can occur.

To find the best suiting representation for visualizing the difference graph, three different questions are relevant rating, intuitive understanding and ranking. This questions showed that three representations always stand on top. Therefore the first selection is that „Symbols“, „Edge Node Font Color“ and „Font Background Color“ are considered for an deeper view.

Both „Symbols“ and „Edge Node Font Color“ match for top place while „Font Background Color“ is behind „Edge Node Font Color“ in every evaluation. This observation also underlay the comments where two attendees mentioned that the understanding is nearly the same but „Edge Node Font Color“ is more memorable and prettier to look at. Therefore „Font Background Color“ will no longer be considered for the implementation.

Selecting the better suiting representation between „Symbols“ and „Edge Node Font Color“ is hard. Symbols are more intuitive to understand and also where on top of the individual ratings. On the other side after seeing every representation, „Edge Node Font Color“ had a better ranking. The evaluation also showed that there are two areas of application for the two representations. „Edge Node Font Color“ suits better for big graphs and „Symbols“ was chosen more often for small ones.

In the comments area the use of a legend was suggested to help users to understand the meaning of each marking. None of the representations had an intuitive understanding of 100% and therefore a legend will be implemented in the prototype implementation. This should secure an uniform understanding and avoid misinterpretation of the difference graph. Depending on the input, the resulting difference graph can be a small graph or a big graph. Therefore an approach which fits both is needed. „Edge Node Font Color“ does not loose any information when it is applied on a small graph. Every color can be expressed on small graphs. On the other side „Symbols“ can be applied to big graphs without having technical problems. However readability within a graph with many nodes and edges can be hard for example, when the graph is to overcrowded and needs to be scrolled the symbol for a specific edge can be invisible because it is not in the point of view and the user needs to scroll. Another example is a graph with many eventually overlapping edges, in such a case, following an edge through the graph to find the symbol can be hard.

Due to the close result finding the best representation is a very difficult task. „Edge Node Font Color“ will be considered for the prototype implementation. The top ranking after seeing every representation and that this representation can be applied to small and big graphs was decisive. With the help of the above mentioned legend the intuitive understanding of „Edge Node Font Color“ will be improved. In the further thesis „Edge Node Font Color“ color will be called color coded representation.

# 7 Implementation Preparation

In the domain of process mining a wide area of commercial and non commercial products is available. For the implementation a suitable environment has to be found. This environment can be either a plug-in for an existing product or a stand alone solution. Section 7.1 covers why a plug-in was preferred. To use a plug-in impacts the choice which mining algorithms should be considered for the implementation. By implementing a plug-in the mining algorithms available in the tool of choice can be used to mine the process models. The following sections will also discuss which mining algorithms will be used.

## 7.1 Process Mining Tool

There are two approaches how to implement the difference graph. Developing a plug-in for an existing tool or develop a standalone tool. In comparison developing a standalone tool needs more effort then developing a plug-in. This conclusion is simple, by developing a plug-in some initial steps like reading/converting different log files and implementing mining algorithms have not to be considered. Developing a plug-in also comes with the advantage that users which already use the software do not need to use a new program they are not familiar with.

In their article [27] list 11 process mining tools which will be analyzed to find the best suiting for the plug-in implementation. Only 4 of the 11 tools are free of charge. It is necessary that the tool is free of charge. This allows to distribute the plug-in to interested persons without the need of purchasing the software. The remaining four candidates are „Genet“ [17], „ProM“ [29], „Rbminer“ [24] and „ServiceMosaic“.

**Genet** is a simple tool which only allows mining Petri Nets from transition systems. It needs a specific input type format and does not offer a built in visualization component.

**ProM** offers exactly what is needed to develop a plug-in. It allows many different input file formats. Offers a modular architecture which enables the use of other plug-ins, in the case of this thesis, mining plug-ins. The developed plug-in can either be placed within the plug-in folder in the installation directory or distributed over the built in plug-in service.

**Rbminer** also mines Petri Nets from transition systems and needs a specific input type format. The output is a text file which can not be visualized as a Petri Net with built in components.

**ServiceMosaic** unfortunately there was nothing to find about this project.

**Selection of Prom** ProM offers everything needed for plug-in development. A modular architecture which does not only enable developing and integrating a plug-in it also allows to use other developed plug-ins. For example the alpha mining algorithm can be used to mine a Petri Net. The Petri Net then can be used for difference calculation. This procedure offers the advantage that the mining function has not to be implemented in the prototype.

### 7.2 Mining Algorithms

ProM supports user generated plug-ins. Most of the available mining algorithms are user generated. There are many different mining algorithms, the goal of the prototype implementation is not to implement every one of them. The goal is to implement a wide spread of different algorithms. This can be achieved by using one algorithm from each category stated in section 2.2 on page 7. This section also shows with which problems the mining algorithms have to deal and how it affects the difference graph. In order to use the output of the mining algorithms they have to fulfill some requirements, like naming, which are also stated in the following sections.

The implementation will feature two different types of usage. One is to use the mining algorithms directly implemented within the plug-in. The other is to mine a process model and use the process model as input for the plug-in.

In [28] the authors state that the three most used algorithms are the Heuristics miner, Genetic miner and Fuzzy miner. None of them exports a Petri Net or a process model without weights. For the implementation one algorithm which does not consist of weights is needed. Therefore the list is extended with the Alpha-algorithm. An overview over the four algorithms considered for the implementation gives Table 8. Every algorithms category, output format and building approach is stated. The output format stated in the table is taken from the PROM implementation of the algorithms. The approach how the process model is generated can be local or global.

| Algorithm | Category | Output | Approach |
|---|---|---|---|
| Alpha-algorithm | Algorithmic | Petri Net | Local |
| Heuristics Miner | Algorithmic | Heuristics Net | Local |
| Fuzzy Miner | Algorithmic | Fuzzy Net | Global |
| Genetic Miner | Genetic | Heuristics Net | Global |

Table 8: This table shows for different mining algorithms how they are categorized, their output format and which approach is used to build the process model.

**Region Based Algorithm?** For mining a log with the use of the theory of regions two plug-ins have to be executed. Thirst a plug-in which generates a transition system and second a plug-in which converts the transition system into a Petri Net. The before mentioned alpha algorithm also exports a Petri Net therefore the difference tool already allows Petri Nets as input. Therefore it is possible to calculate the difference between two process models generated by region based algorithms without implementing further algorithms.

**Miner Output Requirements** As stated in section 5.1 on page 20 a process model needs to fulfill some requirements in order to be able to use it for the

difference graph calculation. The before mentioned mining algorithms export their process models with different formats. For example as Petri Net, Heuristics Net or Fuzzy Net. These formats have to fulfill several requirements to be able to use the mining algorithm which generates them.

**Process Model Output** The output of the miner has to be a process model with nodes and edges. The data stored within the model has to be accessible. For example it is not sufficient when an algorithm exports a process model as an image. To access the nodes an unique label for each activity is necessary. Every activity except from start and end activity have to be connected with input and output edges. Furthermore has every activity to be on a path between start and end node.

**No Clustering** For the difference graph two inputs are necessary. Building cluster of two logs can lead to different problems. For example, both input models can lead to a cluster with same label but the nodes clustered are different. Another example, different labels for the cluster but some nodes are clustered in both cluster.

Figure 16 shows the problems, because of these problems. On the left a cluster is with nodes „A“ and „B“ is built and on the right a cluster for nodes „C“ and „D“ and nodes „A“, „B“ and „C“ is built.

„Clustering same name“ shows the first mentioned example the name of both cluster is the same but the nodes clustered are different. Calculating the difference graph from these two models will lead to a graph where nodes „A“ and „B“ are marked as deleted and nodes „C“ and „D“ as new.

„Clustering different names“ in this example the clustered nodes are part of the label and therefore the label is different. The problem is that some nodes are clustered in both cluster. The overlapping nodes between the two cluster are „A“ and „B“ only node „C“ which is clustered in the blue cluster is different. The difference graph of this cluster will mark the cluster „ABC“ as deleted, the cluster „AB“ as new and the node „C“ also as new.

These problems show that it is not possible to calculate the difference of two process models when clustering is involved.

**Node Types** Mostly the miners visualize the process model with just one node type. However when an output has different node types it is important to know which node is expressed by which type. For example, when the output is a Petri Net it is important to know which node is a place and which a transition. On one side it is important to know this to visualize the process model correctly. On the other side calculating the difference with different node types can get complex. For example, two input process model express one node with the same label but with different node types. One expresses the node as And-Split and the other as Xor-Split. How to calculate the difference between these two? How to visualize the difference? These are unanswered questions and need further investigation.

In this thesis only the Petri Net expresses different node types. A Petri Net consists only of two different node types places and transitions. Places are labeled according to the activities from the log file. Transitions connect places and are

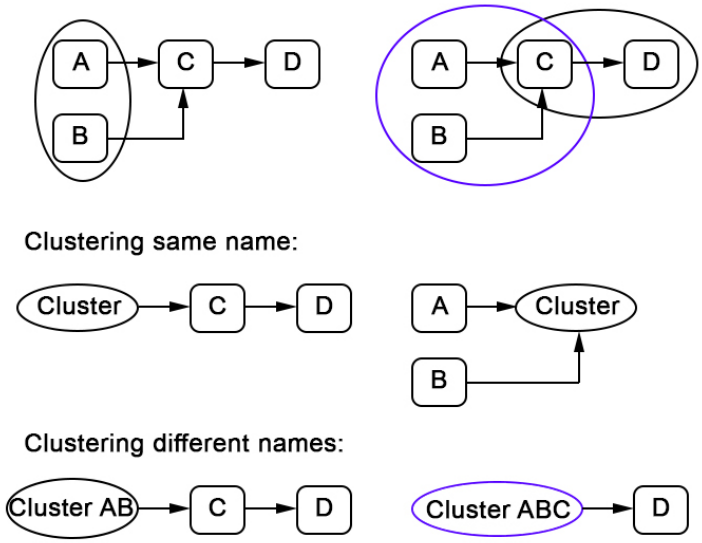



Figure 16: Clustering of data can lead to cluster which have different nodes clustered with same label or different label with same nodes. Both are problematic for calculating the difference.

labeled with the name of the connected places. For this reason even if a place and a transition inherit the same label they are treated differently. An example shows Figure 17 where one place and one transition inherit the label „AB“. In the difference graph both are shown because they express different behavior. Only place „C“ is part of both inputs. The two inputs do not consist of overlapping transitions.

**Selection of Mining Algorithms** With these requirements it is possible to preselect some miners. The Alpha Miner, Heuristics Miner and Genetic Miner fulfill all of the requirements. By simplifying the visualization for the user by clustering the data, building the difference between two Fuzzy Nets can be very complex and therefore it is not part of the implementation.

**Mining Problems do not Matter** Process mining algorithms have to deal with some problems e.g. duplicate activities, hidden activities, non-free-choice constructs, noise and more, details about this problems and how to handle them are stated in [55, 56]. Every mining algorithm which discovers a process model is able to handle these problems as good as possible. However, it is possible that problems were not solved by the mining algorithm, for example, the alpha and heuristics miner can not deal with non-free-choice constructs. Therefore the

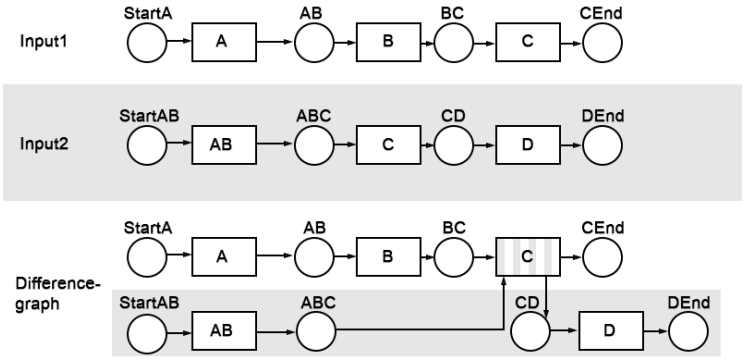



Figure 17: Difference calculation with Petri Nets. Transition „AB“ from „Input1“ and place „AB“ from „Input2“ are treated in a different way there is no difference calculation between these two even if the label is the same.

difference algorithm only compares the input from the same miner and does not mix them up. For that reason the difference analysis plug-in does not consider these problems while calculating the new model.

### 7.3 IDE

ProM was developed with Eclipse [14]. On the project page for ProM developers [30] a tutorial shows how to set up Eclipse and which java version should be installed. After installing Eclipse the tutorial demonstrates how to check out the ProM source code from the repository using Subclipse [23].

Based on these tutorials Eclipse was installed and configured to start developing a plug-in for ProM.

## 8 Implementation and How to Use

This section is divided in 6 steps (8.1 to 8.6). Each step offers a description what is done in the step. Afterwards most of the steps consist of an implementation section where details how the before described concept is implemented are discussed.

### 8.1 Selecting the Input

As mentioned before in Section 7.2 on page 40 the plug-in features two ways to start it.

One way is to run the plug-in with log files. Using log files opens the wizard which is step 2 of this tutorial. The wizard allows a selection of a mining plug-in as well as selecting the time span which should be mined for each log.

The other way is to mine the log files with plug-ins available in ProM and use their output as input for the difference graph plug-in. The difference graph plug-in requires one of the following inputs: Heuristics Net, Petri Net or Activity Nets. Heuristics and Petri Nets are output formats from different mining plug-ins available within the ProM framework. When a miner does not export to this format it may be possible to convert the output into one of these formats. The Activity Net is a class for representing a process model. The Activity Net is the internal representation of a process model of this implementation. The decision to also allow Activity Nets as input was made to secure that other plug-ins can use the plug-in directly without any additional computation power needed for converting the output. An example, a plug-in exports a Fuzzy Net. Another plug-in offers to convert the Fuzzy Net either to a Heuristics Net or an Activity Net. Both can be used as input for the difference graph plug-in. The only difference between these two is that within the difference graph plug-in the Heuristics Net is converted into the internal representation which is a Activity Net. After generating/converting the input into one of the accepted formats the difference graph plug-in can be started. When the plug-in is started with two already mined models the wizard will be skipped and step 3 is executed.

### 8.2 Wizard

After selecting the log files and opening the difference graph tool the wizard is started. The wizard allows to select one of three mining algorithms. These are the Heuristics, Genetic and Alpha Miner. A trace picker displayed as a slider allows to select start and end trace for each input file. To enhance useability the trace picker offers additional information, for example, which traces are selected, how many traces are selected and start plus end date of the selection.

Running the plug-in with log files also allows to run the plug-in with the same log file as input1 and input2. Normally this can not produce any differences because the difference between two times the same input is zero. However, the wizard allows to select the time span which should be mined. For example, a log file consists of traces for two year from start of 2007 until the end of 2008. For the first input the whole year 2007 is picked and for the second input the year 2008 is picked. This allows to see the differences between these two years. Therefore the plug-in does not only show the differences of distinct processes it is also capable of showing how a process has evolved.

Figure 18 shows that from the first input 43 traces are used. Start date is 2006.01.01 and end date 2006.12.23. From the second 41 traces for the year 2007 are used. This shows the first differences between the years. In year 2006 the process was executed 43 times while in 2007 it was executed 41 times.

**Implementation** The ProM Framework allows to generate this kind of wizard with build in methods. The built in wizard is used to secure a consistent look and feel within the framework. The wizard method offers everything needed and therefore it was used. The method generates the look and handling, for example,

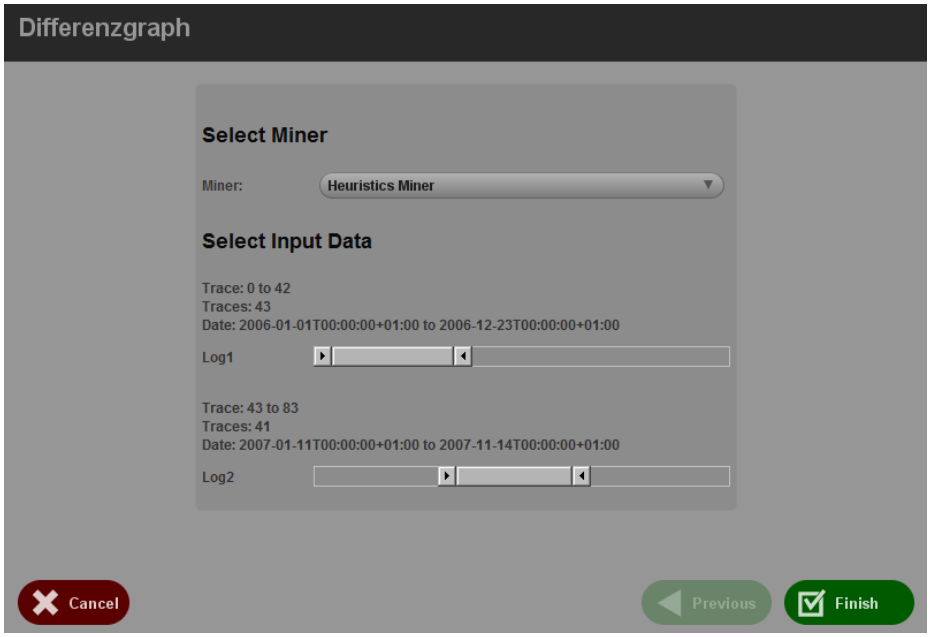



Figure 18: The wizard is executed each time log files are used as input for the difference graph. The wizard allows selection of mining plug-ins and time span which should be mined for each log.

the buttons on bottom. The wizard method does not generate fields like miner selection or trace picker.

The miner selection is a combo box styled with the „SlickerFactory“ [11]. The „SlickerFactory“ is a library which allows to style certain elements automatically that they fit into the ProM design and is part of the UIGuidelines [31].

JRangeSlider [20] is used to implement the trace picker for both input log files. The trace picker is useless without chronological ordered traces and this is not a requirement for log files. Therefore before the trace picker can be filled with data, the log file has to be ordered. To do so, the log files where ordered by start date for each trace. To sort the log the Quicksort algorithm [10] was chosen for implementation.

### 8.3 Internal Representation

After data and miner selection the log files are mined. To mine the logs the chosen algorithm is called once for each of the logs. These calls return different models depending on the chosen algorithm, for example, Petri Nets or Heuristics Nets. This brings together the approaches to use the wizard and to use mined models which where stated in Section 8.1 on page 43.

From here on there are two possible ways. The first is to work with these models as they come. The second is to transform the models into an other model. In this thesis the transformed model is called Activity Net. To use the models directly has the advantage that no transformation is needed. This approach would be very good if there is only one model type. As mentioned before the mining algorithms return different model types and therefore the decision to transform the models was made. Transforming the models leads to several advantages:

**Easy extension** One advantage is that the tool can be extended easily. To add new model types the only thing to do is to add a transformation routine how the new model type is transformed into an Activity Net.

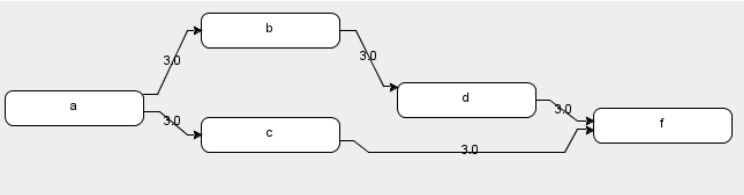
**One difference calculation** Another advantage is that the difference calculation is implemented based on the Activity Net. Without the Activity Net the difference calculation has to be implemented for each model type.

**One visualization** Every of the model types offers a specific visualization. Using these visualizations within the difference graph needs to meet several design choices mentioned in Section 5.2 on page 21. Therefore every visualization needs to be adapted to meet the design choices. With the Activity Net the visualization has to be implemented only once to meet these design choices.

**Modular concept** Overall using an internal model leads to a modular program where individual parts can be changed independent from the rest of the program. For example changing the calculation routine of the difference will not affect the visualization component.

An Activity Net is represented by a new class which consists of a matrix and a string array. The matrix is an adjacency matrix which consists of weights for each edge. The matrix is not symmetric because the graph is a directed graph. The string array represents the unique names of the nodes. Figure 19 shows on the left a visualization of a Heuristics Net and on the right the visualization

transformed into an adjacency matrix. The string array is shown on top and left of the matrix. Every zero in the matrix represents that there is no edge between two nodes. Values distinct from zero represent the corresponding weight from the visualization.




| | a | b | c | d | f |
|---|---|---|---|---|---|
| a | 0 | 3 | 3 | 0 | 0 |
| b | 0 | 0 | 0 | 3 | 0 |
| c | 0 | 0 | 0 | 0 | 3 |
| d | 0 | 0 | 0 | 0 | 3 |
| f | 0 | 0 | 0 | 0 | 0 |

Figure 19: The Heuristics Net on the left is transformed into the adjacency matrix shown on the right.

### 8.4 Calculation of the Difference Graph

Before the difference can be calculated the input has to be prepared. There are several things to consider by the calculation and therefore just input1 minus input2 is not practicable. This section first shows the concept which things have to be considered to perform the difference calculation. Afterwards the implementation is described and where it differs from the concept.

**Conceptual Considerations** The following paragraphs show conceptual thoughts what has to be considered before the difference calculation can be done. The implementation distinguishes from these considerations because the steps are tailored for performance optimization.

**Same Order of Activities** Even if the log files are identical it is possible that activities have an alternate order when the process is mined. Once the process model is transformed to an adjacency matrices it is important that both matrices have the same order of activities otherwise the difference graph calculation will be incorrect. Therefore one matrix has to be adapted to the other. Figure 20 shows how this problem can look and how to solve such a problem. In the example „Matrix 2“ is adapted to „Matrix 1“.

An important note at this point is that for the difference graph it is not relevant which matrix has to be adapted. Also the order which activity is first and which is last in the matrix is not important for the difference calculation as long as it is the same order in both matrices. Only the calculation needs the same order of activities. The visualization component does not pick the first row or column to draw first. The visualization algorithm searches in the matrix for the start point/s and draws it/them first.

**Deal with New or Deleted Activities** When a log contains activities which the other does not contain, the adjacency matrices can be different in

| Matrix 1 | Initialize | Process Data | Call Miner | Rendering |
|---|---|---|---|---|
| Initialize | 0 | 2 | 1 | 0 |
| Process Data | 0 | 0 | 2 | 0 |
| Call Miner | 0 | 0 | 0 | 3 |
| Rendering | 0 | 0 | 0 | 0 |

| Matrix 2 | Initialize | Call Miner | Process Data | Rendering |
|---|---|---|---|---|
| Initialize | 0 | 0 | 3 | 0 |
| Call Miner | 0 | 0 | 0 | 1 |
| Process Data | 0 | 1 | 0 | 2 |
| Rendering | 0 | 0 | 0 | 0 |

| Solution | Initialize | Process Data | Call Miner | Rendering |
|---|---|---|---|---|
| Initialize | 0 | 3 | 0 | 0 |
| Process Data | 0 | 0 | 1 | 2 |
| Call Miner | 0 | 0 | 0 | 1 |
| Rendering | 0 | 0 | 0 | 0 |

Figure 20: „Matrix 1“ shows a different order of activities as shown in „Matrix 2“, by swapping the columns and rows „call miner“ and „process data“ from „Matrix 2“, the logs will have the same order and the difference can be calculated.

size, figure 21 shows an example. Calculating the difference of two different sized matrices can be very complex. A solution to this problem is when ever a matrix contains an activity which the other does not contain this activity will be inserted with zero weight values.

**Additional Matrix for Change** The evaluation showed that the color coded version will be implemented. To be able to assign the correct color to each node and edge it is important to exactly know what has changed. There are five different types of change which can be calculated if weights are given new, positive changed, unchanged, negative changed and deleted and three types of change new, unchanged and deleted when no weights are given.

There are two ways to visualize the correct style while drawing the difference graph. One is to store the change operation for each edge while the difference graph is calculated, for example, in a matrix. The other is to calculate the style while the difference graph is drawn. The MVC architecture [39] is used and therefore calculating while drawing is no suitable answer. Furthermore why calculate the style again when it was already calculated. Therefore the style is stored in the same way the weights are stored, within an adjacency matrix. Figure 22 at section 8.4 on page 48 shows how such a matrix looks. With this matrix drawing the difference graph is easy. When an edge is drawn the corresponding value from the change matrix will be read and shows which style should be used to draw this edge, for example, when the change matrix states the edge is marked as deleted the color will be red.

| **Matrix 1** | Initialize | Process Data | Call Miner | Rendering |
|---|---|---|---|---|
| Initialize | 0 | 2 | 1 | 0 |
| Process Data | 0 | 0 | 2 | 0 |
| Call Miner | 0 | 0 | 0 | 3 |
| Rendering | 0 | 0 | 0 | 0 |

| **Matrix 2** | Initialize | Call Miner | Rendering |
|---|---|---|---|
| Initialize | 0 | 1 | 0 |
| Call Miner | 0 | 0 | 1 |
| Rendering | 0 | 0 | 0 |

| **Solution** | Initialize | **Process Data** | Call Miner | Rendering |
|---|---|---|---|---|
| Initialize | 0 | **0** | 1 | 0 |
| **Process Data** | **0** | **0** | **0** | **0** |
| Call Miner | 0 | **0** | 0 | 1 |
| Rendering | 0 | **0** | 0 | 0 |

Figure 21: Log two consists of lesser activities then log one because activity „Process Data“ does not exist. To be able to calculate the difference the missing activity is inserted with zero values.

**Calculate the Difference** When both adjacency matrices have the same size and order it is easy to calculate the difference. Just matrix2 minus matrix1 the result is a new matrix which represents the difference graph. To support the visualization process an additional matrix for change operations will be used.

Figure 22 will give an example for difference calculation and a change matrix where every marking is computed at least once. The following list gives details how the markings stated in Section 3.1 on page 9 are calculated when a matrix is given. In this case the input models consist of weights and therefore the list shows an example for five markings. When the input process models do not contain weights only three markings can be calculated and stored within the change matrix. The two markings „Positively changed“ and „Negatively changed“ can not be calculated without weights.

- **New**, an edge is new if the edge in matrix1 has a zero and in matrix2 has an other value. (process data to rendering)
- **Positively changed**, an edge is positively changed if in matrix1 it has a number and in matrix2 it has higher number. (initialize to call miner)
- **Unchanged**, an edge is unchanged if it has the same weight in matrix 1 and matrix 2. (call miner to rendering)
- **Negatively changed**, an edge is negatively changed if in matrix1 it has a number and in matrix2 it has lower number. (edge from initialize to process data)
- **Deleted**, an edge is deleted if the edge in matrix 1 has a value distinct from zero and in matrix 2 a zero. (edge from process data to call miner)
- If an edge has zero values in both matrices no entry is made in the change matrix.

| Matrix 1 | Initialize | Process Data | Call Miner | Rendering |
|---|---|---|---|---|
| Initialize | 0 | 2 | 1 | 0 |
| Process Data | 0 | 0 | 2 | 0 |
| Call Miner | 0 | 0 | 0 | 3 |
| Rendering | 0 | 0 | 0 | 0 |

| Matrix 2 | Initialize | Process Data | Call Miner | Rendering |
|---|---|---|---|---|
| Initialize | 0 | 1 | 3 | 0 |
| Process Data | 0 | 0 | 0 | 1 |
| Call Miner | 0 | 0 | 0 | 3 |
| Rendering | 0 | 0 | 0 | 0 |

| Difference | Initialize | Process Data | Call Miner | Rendering |
|---|---|---|---|---|
| Initialize | 0 | -1 | 2 | 0 |
| Process Data | 0 | 0 | 2 | 1 |
| Call Miner | 0 | 0 | 0 | 0 |
| Rendering | 0 | 0 | 0 | 0 |

| Changes | Initialize | Process Data | Call Miner | Rendering |
|---|---|---|---|---|
| Initialize | | negatively changed | positively changed | |
| Process Data | | | Deleted | New |
| Call Miner | | | | Unchanged |
| Rendering | | | | |

Figure 22: From Matrix1 and Matrix2 the difference and change matrix are calculated. The calculation was Matrix2 minus Matrix1.

**Implementation** As mentioned before the implementation differs from the conceptual considerations.

First the concept „Deal with New or Deleted Activities“ is addressed. In the concept the input matrices are changed, not changing the input matrices gains performance in comparison to changing both matrices. This is done by merging only the labels of the two inputs and then generating two empty matrices with the length of the merged labels. Figure 23 and Algorithm 1 show all steps which are performed for the difference calculation. Merging the labels is Step1, generating two empty matrices Step2. In the further one of these matrices is called difference matrix and the other change matrix. The difference matrix includes the values calculated by the difference calculation. While the change matrix holds the markings assigned to each edge.

The second concept „Same Order of Activities“ also changes the input matrices which needs performance to swap rows and columns to have equal matrices. In the implementation the order of the before merged labels is used. They are not rearranged or sorted the order is used as it comes.

As Mentioned before calculating the difference with different matrix size can be complex. In this case the size of the output matrix and row/column labels are given. The third step in the implementation is to insert every value from matrix2 into the difference matrix.

The fourth step is the calculation of the difference. Therefore the values from matrix1 are subtracted from the values stored in the difference matrix. The result of this calculation overrides the value stored within the difference matrix.

In this step the change matrix is filled. Which marking is assigned to which edge is decided by the values from the difference matrix and matrix1. The list from Section 8.4 on page 48 gives details which markings are possible and how they are calculated.


Figure 23: The input for the difference calculation are two matrices. On the right of the matrices the corresponding graph is visualized. Those two matrices are used as input for the difference calculation which is divided in four steps and results in a difference and change matrix. The result is also visualized as color coded graph.

The next section focuses on visualizing the graph. To do this the difference and change matrix are needed.

**Algorithm 1** Calculation algorithm

```
{Step1}
Input1Lables = GetLablesFromInput1();
Input2Lables = GetLablesFromInput2();
DiffGraphLables = Merge(Input1Lables, Input2Lables);
{Step2}
DiffMatrix = NewMatrix(Length(DiffGraphLables), Length(DiffGraphLables));
ChangeMatrix = Copy(DifferenceMatrix);
{Step3}
for In2 = Every Edge From Input2 do
    for Diff = Every Edge From DiffMatrix do
        if GetLabel(In2) equals GetLable(Diff) then
            Diff = In2;
        end if
    end for
end for
{Step4}
for In1 = Every Edge From Input1 do
    for Diff = Every Edge From DiffMatrix do
        if GetLabel(In1) == GetLable(Diff) then
            {Example for New Node}
            if Value(Diff) != 0 AND Value(In1) == 0  then
                ChangeMatrix.Insert(GetPosition(Diff), „New“)
            end if
            {Overwrite old, with new weight}
            Value(Diff) = Value(Diff) - Value(In1);
        end if
    end for
end for
```

## 8.5 Visualization of the Difference Graph

Figure 24 shows the visualization of the inputs and difference graph mined as Heuristics Net. As mentioned in the design choices in Section 5.2 on page 21 the three containers match their space 20%/20%/60%. For each input 20% of the space within the window is reserved. The two inputs are placed side by side. On bottom of them with a space of 60% the difference graph is displayed. The difference graph is visualized with color coding which was evaluated as the best way to show the differences between the inputs.

- **New** nodes and edges are colored green.
- **Positively changed** nodes and edges are displayed blue.
- **Unchanged** nodes and edges are visualized black.
- **Negatively changed** nodes and edges are shown as orange.
- **Deleted** nodes and edges are visualized red.

The same input log files which where taken to mine Figure 24 where taken as input for Figure 25. The difference is that the inputs of Figure 25 where mined with the Alpha Miner. The Output of the Alpha Miner is a Petri Net.

As mentioned earlier in the design choices the nodes and edges are positioned on the same coordinates in every of the visualizations. For Example node „A“ is positioned on the left in each graph while node „D“ is positioned on the right. This also changes what can be seen within the difference graph. In the Petri Net there are no weights on the edges. No weights means no way to visualize positively and negatively changes. While in the Heuristics Net the Nodes „A“ and „B“ where marked as negatively changed this marking is not possible within the Petri Net. In the Petri Net the nodes are marked as unchanged.

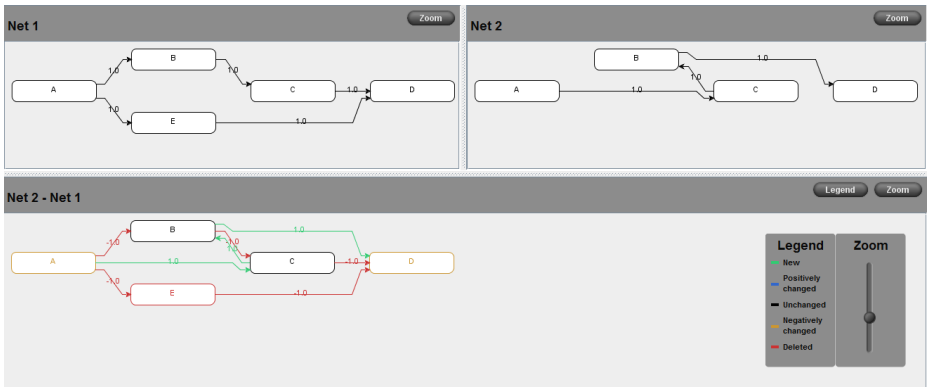



Figure 24: Screenshot of the Difference graph plug-in. Both inputs are visualized on top and on bottom the difference graph is visualized. The difference graph is visualized with color coded nodes and edges as result from the evaluation.

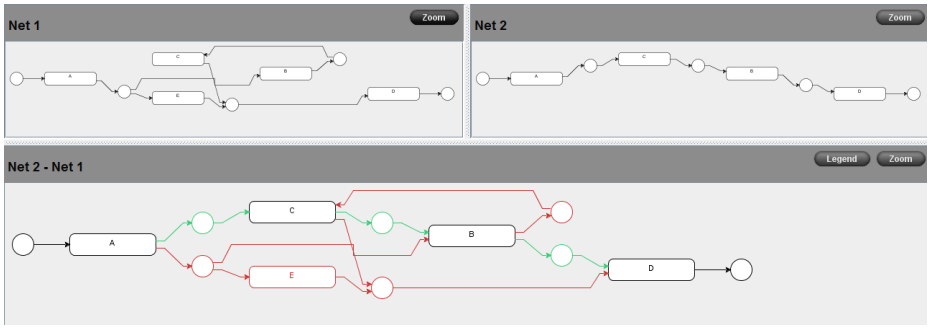



Figure 25: Screenshot of the Difference graph plug-in. The Screenshot shows two input Petri Nets. The difference graph can only visualize three different markings (New (Green), Red (Deleted), Black (Unchanged).

**Implementation** The positioning of the graphs is done with SplitPanes. Besides the positioning side by side and the difference graph on bottom. SplitPanes allow to move the border which is used to divide the graphs. By moving the border the user is able to re-size the graphs to his needs.

Based on the design choices nodes and edges should be placed on the same position over all three representations. This leads to two possible ways. One is to draw the input first and then the difference graph. The other is to draw the difference graph and based on it the input.

By drawing one input and based on the first input draw the second one has a minor disadvantage. None of both inputs consists of every node and edge which can be displayed in the other input. Imagine the following example. The first graph is drawn with nodes in one row „B“ „C“ „D“. Based on the first graph the second graph is drawn. The second graph consists of a sequence „A“ „B“ „C“ „D“. The node „A“ should be inserted before node „B“ to do so both graphs have to be re-arranged. This is just a simple example with more nodes and edges this can get a challenging task.

To simplify this the first graph to draw can be the difference graph. The difference graph consists of every node and edge available in both inputs and therefore the difference graph is perfect for drawing first. Based on the difference graph the two inputs are drawn. Algorithm 2 shows how the visualization component was implemented.

**How the difference graph is drawn.** For visualizing the difference graph the library „JgraphX“ [19] is used. An object from type „mxGraph“ is instantiated. Afterwards nodes can be added by using the function „insertVertex“ the most important parameters are label and style of the node. The standard style is a rectangle and is only changed when the graph is a Petri Net. Within a Petri Net two styles are possible a rectangle for transitions and a circle for places. The style also consists of the color which represents the marking applied to the node. After inserting the nodes the connections between them have to be inserted. To

do so the function „InsertEdge“ is used. It´s prime parameters are the label of the node which is the calculated weight of a node (stored within the difference matrix). As well as start and end node of the edge and the style which should be applied to the node. After filling the „mxGraph“ object with nodes and edges the graph has to be drawn. To draw a graph the library offers layout algorithms. A control-flow related layout algorithm is the „mxHierarchicalLayout“ and therefore this algorithm was used.

**How the input graphs are drawn.** To draw the input graphs two ways are possible. One is to build a new graph and position the nodes and edges in the way they are positioned in the difference graph. The other is to copy the difference graph and remove nodes and edges which are not displayed in the input. The approach to remove not used nodes and edges was chosen. This approach offers that without manually tweaking the graph it looks exactly the same as the difference graph. Using this approach it is important to first remove the unused edges and then remove unused nodes. The reason for this is that edges are defined by the nodes they are connecting. When a node is deleted the edge can not be selected because the node is not accessible any more. After removing the edges which should not be displayed. The labels of the remaining edges are overwritten by the values stored within the corresponding input matrix. The style of the edges is overwritten by a black edge style. Before drawing the input unused nodes have to be removed and the color of the remaining nodes has to be set to black.

### 8.6 User Interactions

To enhance the usability additional functions are added to the difference graph. The functions zoom, pan and brushing are declared as standard functions by [9]. Based on the evaluation a legend will also be implemented. The functions zoom and legend will be accessible with buttons. These two functions will not be visible permanently. To show and hide these functions with a button should prevent permanent overlapping of the graph. Figure 26 shows an example where the overlay for zoom and legend are displayed within the difference graph. To show all nodes and edges from the two inputs they are zoomed out. The nodes „b“, „c“ and „d“ where selected in the difference graph. This selection is also applied to the two input graphs.

**Zooming**, allows to zoom in and out of the graph. This function can be helpful by small as well as big graphs.

**Panning**, allows to move the graph in vertical and horizontal direction.

**Brushing**, allows to select a node. A selected node is visualized by a different background color. To better fit the difference graph visualization the brushing function was extended. When one node is selected in any graph the corresponding node is also selected in the other graphs. To automatically select the corresponding node in every graph should secure a faster understanding of the difference graph. As shown in Figure 26 brushing also allows to select multiple nodes.

**Algorithm 2** Visualization algorithm

```
1: graph = new mxGraph();
2: {Insert Nodes}
3: for DiffNodes = Every node from the Difference graph do
4:    if DifferenceGraph == PetriNet then
5:       if DiffNodes == Place then
6:          graph.insertVertex(GetLabel(DiffLabel), Circle, GetCo-
            lor(GetMarking(DiffNodes)));
7:       else
8:          graph.insertVertex(GetLabel(DiffLabel), Rectangle, GetCo-
            lor(GetMarking(DiffNodes)));
9:       end if
10:   else
11:      graph.insertVertex(GetLabel(DiffLabel), Rectangle, GetCo-
         lor(GetMarking(DiffNodes)));
12:   end if
13: end for
14: {Insert Edges}
15: for DiffEdges = Every edge from the Difference graph do
16:   if Edge != 0 or Marking == Unchanged then
17:      graph.insertEdge(GetLable(DiffEdges), GetFrom(DiffEdges), Get-
         To(DiffEdges), GetColor(GetMarkingFromMatrix(DiffEdges)));
18:   end if
19: end for
20: {Apply Layout}
21: Layout = new mxHierarchicalLayout();
22: graph.execute(Layout);
23: {Generate Input1}
24: Input1Graph = Copy(graph);
25: {Remove Edges}
26: for EdgeDiff = Every Edge From Input1Graph do
27:   DeleteEdge = true;
28:   for EdgeInput1 = Every Edge From Input1 Matrix do
29:      if EdgeInput1 == EdgeDiff then
30:         DeleteEdge = false;
31:         EdgeDiff.Style(black, GetLabel(EdgeInput1));
32:      end if
33:   end for
34:   if DeleteEdge == true then
35:      graph.remove(EdgeDiff)
36:   end if
37: end for
38: {Remove Nodes}
39: for NodeDiff = Every Node From Input1Graph do
40:   DeleteNode = true;
41:   for NodeInput1 = Every Edge From Input1 String Array do
42:      if NodeInput1 == NodeDiff then
43:         DeleteNode = false;
44:         NodeDiff.Style(black, GetLabel(NodeInput1));
45:      end if
46:   end for
47:   if DeleteNode == true then
48:      graph.remove(NodeDiff)
49:   end if
50: end for
51: Do the same for Input2
```

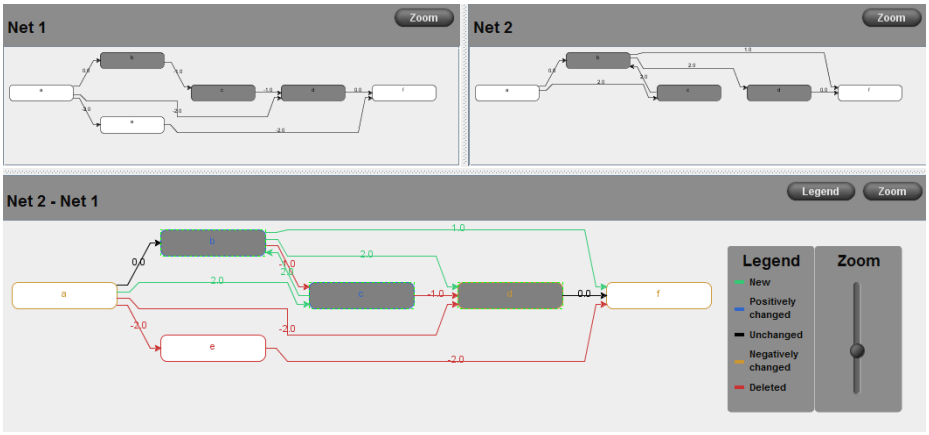



Figure 26: Screenshot of the difference graph plug-in. Both inputs are zoomed out and the selected nodes from the difference graph are also selected in the inputs.

**Legend**, the legend will only be available within the difference graph visualization. The legend describes either five or three different styles which are visualized through color coding.

**Implementation** For both the zoom and legend function a „JButton“ with an „ActionListener“ was generated. After click on the button the function is displayed. This works based on a multilayer architecture. On the bottom layer the graph is visualized. One layer above the functions zoom an legend are visualized. Both functions are visualized through „JPanels“. The „JPanel“ for the legend consists of a „TableLayout“ which on the left visualizes the style and on the right the label. Every „JPanel“ is styled with the „SlickerFactory“ to allow a seamless integration to the ProM design. The „JPanel“ for the zoom function consists of a styled slider with a minimum value of 10, an origin of 100 and a maximum value of 250. This means that the view can be zoomed by a factor of 2.5 and zoomed out by a factor of 10.

Within the „jgraphx“ library a listener is available which listens for selected cells. A cell can be a node, edge, label or some other things. This listener is used for the brushing function. Each time the listener is activated a request checks if the selection is a node. When the selection is a node the background color of the old selection is set to white. Afterwards the labels of the selection are pushed to all graphs which perform a selection based on the labels. When a label is not available within one graph the label is skipped. A selection is visualized by changing the background color of the nodes to grey.

## 9 Future Work

This thesis has answered several questions how to visualize the differences between two process models. However, there are still answers which have to be found. In terms of visualization a more detailed evaluation comparing the two styles symbols and color coding can be helpful. This can help to secure the graph is visualized in the way the user understands it the most.

Processes are evolving over time. To visualize this the difference visualization with an input of two process models is not enough. How can the difference calculation compute relevant evolution steps within a process? How can this evolution be visualized?

As mentioned earlier in this thesis there exist many different modeling languages. In this thesis process models consisted of a maximum of two different node types (Petri Nets). What to do when there more are different node types with different semantics? Can the difference graph concept be used within an environment of different nodes?

## 10 Conclusion

The goal of this thesis was to implement a prototype for difference calculation and visualization. The concept of the difference graph was described. Within the difference graph there are up to five markings possible which show the differences between two processes. To find a visualization for these five markings a literature research was done. There was no single outstanding good representation. Therefore an evaluation was conducted. The goal was to find the best suiting representation for the difference graph. To show a working example of the difference graph a plug-in for the ProM Framework was developed. Before specified requirements, design choices and results of the evaluation where involved during the development process.

One of the big accomplishments was do find a representation for the difference graph. The color coded representation had the overall best ranking. The color coded representation uses a different color for each marking. For nodes the color is applied to the border of the shape and the font. For edges the color is applied on the edge and the font.

The second big accomplishment was to bring the concept and survey results together and implement a prototype. The implementation showed that the concept, requirements, design choices and evaluation results can be combined.

With the results and prototype from this thesis comparing two processes got easier.

## Bibliography

1. Lindsay A., Downs D., and Lunn K. Business processes—attempts to find a definition. *Information and Software Technology*, 45(15):1015 – 1019, 2003. Special Issue on Modelling Organisational Processes.

2. Rozinat A. Process mining: conformance and extension. *TU Eindhoven, Diss*, 2010.
3. Schipper A., Fuhrmann H., and von Hanxleden R. Visual comparison of graphical models. In *Engineering of Complex Computer Systems, 2009 14th IEEE International Conference on*, pages 335–340, June 2009.
4. Alves de Medeiros A. K., Weijters A.J.M.M., and van der Aalst W.M.P. Genetic process mining: an experimental evaluation. *Data Mining and Knowledge Discovery*, 14(2):245–304, 2007.
5. Weijters A.J.M.M., van der Aalst W.M.P., and Alves de Medeiros A. K. Process mining with the heuristics miner-algorithm. *Technische Universiteit Eindhoven, Tech. Rep. WP*, 166:1–34, 2006.
6. Schönhage B., van Ballegooij A., and Elliëns A. 3d gadgets for business process visualization—a case study. In *Proceedings of the Fifth Symposium on Virtual Reality Modeling Language (Web3D-VRML)*, VRML '00, pages 131–138, New York, NY, USA, 2000. ACM.
7. van Dongen B. F., Busi N., Pinna G., and van der Aalst W.M.P. *An iterative algorithm for applying the theory of regions in process mining.* Beta, Research School for Operations Management and Logistics, 2007.
8. Schmitz C. Limesurvey. http://www.limesurvey.org/, 2014. [Online; accessed 12-March-2014].
9. Tominski C., Abello J., and Schumann H. Cgv—an interactive graph visualization system. *Computers and Graphics*, 33(6):660 – 678, 2009.
10. Hoare C. A. R. Quicksort. *The Computer Journal*, 5(1):10–16, 1962.
11. Günther C. W. Slickerfactory. http://code.deckfour.org/slickerbox/, 2014. [Online; accessed 27-September-2014].
12. Günther C. W. and van der Aalst W.M.P. Fuzzy mining – adaptive process simplification based on multi-perspective metrics. In *Business Process Management*, volume 4714 of *Lecture Notes in Computer Science*, pages 328–343. Springer Berlin Heidelberg, (2007).
13. Archambault D. Structural differences between two graphs through hierarchies. In *Proceedings of Graphics Interface 2009*, GI '09, pages 87–94, Toronto, Ont., Canada, Canada, 2009. Canadian Information Processing Society.
14. Eclipse. Eclipse. https://eclipse.org/, 2014. [Online; accessed 03-August-2014].
15. Google. Google scholar. http://scholar.google.at/, 2014. [Online; accessed 16-January-2014].
16. Purchase H., Hoggan E., and Görg C. How important is the "mental map"? – an empirical investigation of a dynamic graph layout algorithm. In Michael Kaufmann and Dorothea Wagner, editors, *Graph Drawing*, volume 4372 of *Lecture Notes in Computer Science*, pages 184–195. Springer Berlin Heidelberg, 2007.
17. Carmona J. Genet. http://www.cs.upc.edu/ jcarmona/genet.html, 2014. [Online; accessed 02-August-2014].
18. Carmona J., Cortadella J., and Kishinevsky M. A region-based algorithm for discovering petri nets from event logs. In *Business Process Management*, volume 5240 of *Lecture Notes in Computer Science*, pages 358–373. Springer Berlin Heidelberg, 2008.
19. JgraphX. Jgraphx. https://jgraph.com/, 2014. [Online; accessed 19-September-2014].
20. JRangeSlider. Jrangeslider. https://github.com/prefuse/Prefuse/blob/master/src/prefuse/util/ui/JRangeSlider.java, 2014. [Online; accessed 19-September-2014].

21. Alanen M. and Porres I. Difference and union of models. In Perdita Stevens, Jon Whittle, and Grady Booch, editors, *UML 2003 - The Unified Modeling Language. Modeling Languages and Applications*, volume 2863 of *Lecture Notes in Computer Science*, pages 2–17. Springer Berlin Heidelberg, 2003.
22. Kazuo M., Peter E., Wei L., and Kozo S. Layout adjustment and the mental map. *Journal of Visual Languages and Computing*, 6(2):183 – 210, 1995.
23. Phippard M. Subclipse. http://marketplace.eclipse.org/content/subclipse, 2014. [Online; accessed 06-August-2014].
24. Sole M. rbminer. http://personals.ac.upc.edu/msole/homepage/rbminer.html, 2014. [Online; accessed 02-August-2014].
25. Sebrechts M. M., Cugini J. V., Laskowski S. J., Vasilakis J., and Miller M. S. Visualization of search results: A comparative evaluation of text, 2d, and 3d interfaces. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 3–10, New York, NY, USA, 1999. ACM.
26. Microsoft. Microsoft academic. http://academic.research.microsoft.com/, 2014. [Online; accessed 16-January-2014].
27. Gehrke N. and Werner M. Process mining. *WISU - die Zeitschrift für den Wirtschaftsstudenten*, 2013.
28. Esmita P. G. Process mining a comparative study. *nternational Journal of Advanced Research in Computer and Communication Engineering*, 3, 2014.
29. ProM. Prom. http://www.promtools.org/, 2014. [Online; accessed 02-August-2014].
30. ProM. Prom developer tutorial. https://svn.win.tue.nl/trac/prom/wiki/setup/HowToBecomeAProMDeveloper, 2014. [Online; accessed 03-August-2014].
31. ProM. Uiguidelines. https://svn.win.tue.nl/trac/prom/wiki/UIGuidelines, 2014. [Online; accessed 17-August-2014].
32. Accorsi R., Ullrich M., and van der Aalst W.M.P. Process mining. *Informatik-Spektrum*, 35:354–359, (2012).
33. Agrawal R., Gunopulos D., and Leymann F. Mining process models from workflow logs. In *Advances in Database Technology — EDBT'98*, volume 1377 of *Lecture Notes in Computer Science*, pages 467–483. Springer Berlin Heidelberg, 1998.
34. Dijkman R., Dumas M., and García-Bañuelos L. Graph matching algorithms for business process model similarity search. In *Business Process Management*, volume 5701 of *Lecture Notes in Computer Science*, pages 48–63. Springer Berlin Heidelberg, 2009.
35. Finley R. and Finley C. Microsoft excel. https://www.surveymonkey.com/, 2014. [Online; accessed 12-March-2014].
36. Porst R. Fragebogen. *Ein Arbeitsbuch, Wiesbaden*, 22, 2008.
37. Brown R. A., Recker J. C., and West S. Using virtual worlds for collaborative business process modeling. *Journal of Business Process Management*, 17(3):546–564, 2011.
38. Betz S., Eichhorn D., Hickl S., Klink S., Koschmider A., Li Y., Oberweis A., and Trunko R. 3d representation of business process models. *MobIS*, 8:73–87, 2008.
39. Hansen S. and Fossum T. V. Refactoring model-view-controller. *J. Comput. Sci. Coll.*, 21(1):120–129, October 2005.
40. Jablonski S. and Goetz M. Perspective oriented business process visualization. In *Business Process Management Workshops*, volume 4928 of *Lecture Notes in Computer Science*, pages 144–155. Springer Berlin Heidelberg, 2008.

41. Kabicher S., Kriglstein S., and Rinderle-Ma S. Visual change tracking for business process models. In *Proceedings of the 30th International Conference on Conceptual Modeling*, ER'11, pages 504–513, Berlin, Heidelberg, 2011. Springer-Verlag.
42. Kirchhoff S., Kuhnt S., Lipp P., and Schlawin S. Der fragebogen. *Datenbasis, Konstruktion und Auswertung*, 3:22, 2003.
43. Kriglstein S., Wallner G., and Rinderle-Ma S. A visualization approach for difference analysis of process models and instance traffic. In *Business Process Management*, volume 8094 of *Lecture Notes in Computer Science*, pages 219–226. Springer Berlin Heidelberg, (2013).
44. White S. A. Introduction to bpmn. *IBM Cooperation*, 2(0):0, 2004.
45. Li T., Liang F., Ma S., and Peng W. An integrated framework on mining logs files for computing system management. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 776–781, New York, NY, USA, 2005. ACM.
46. Masui T. Graphic object layout with interactive genetic algorithms. In *Visual Languages, 1992. Proceedings., 1992 IEEE Workshop on*, pages 74–80, Sep 1992.
47. Ullmann T. maq-online. http://ssl.maq-online.de/, 2014. [Online; accessed 12-March-2014].
48. Fruchterman T. M. J. and Reingold E. M. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991.
49. Doğrusöz U., Madden B., and Madden P. Circular layout in the graph layout toolkit. In Stephen North, editor, *Graph Drawing*, volume 1190 of *Lecture Notes in Computer Science*, pages 92–100. Springer Berlin Heidelberg, 1997.
50. van der Aalst W.M.P. Formalization and verification of event-driven process chains. *Information and Software Technology*, 41(10):639 – 650, 1999.
51. van der Aalst W.M.P. Conformance checking. In *Process Mining*, pages 191–213. Springer Berlin Heidelberg, 2011.
52. van der Aalst W.M.P. Process discovery: An introduction. In *Process Mining*, pages 125–156. Springer Berlin Heidelberg, 2011.
53. van der Aalst W.M.P., Alves de Medeiros A. K., and Weijters A.J.M.M. Process equivalence: Comparing two process models based on observed behavior. In *Business Process Management*, volume 4102 of *Lecture Notes in Computer Science*, pages 129–144. Springer Berlin Heidelberg, 2006.
54. van der Aalst W.M.P. and Hofstede A.H.M. Yawl: yet another workflow language. *Information Systems*, 30(4):245 – 275, 2005.
55. van der Aalst W.M.P. and Weijters A.J.M.M. Process mining: a research agenda. *Computers in Industry*, 53:231 – 244, 2004.
56. van der Aalst W.M.P., de Medeirosand A.K. Alves, and Weijters A.J.M.M. Genetic process mining. In Gianfranco Ciardo and Philippe Darondeau, editors, *Applications and Theory of Petri Nets 2005*, volume 3536 of *Lecture Notes in Computer Science*, pages 48–69. Springer Berlin Heidelberg, (2005).
57. van der Aalst W.M.P., Reijers H. A., Weijters A.J.M.M., van Dongen B. F., Alves de Medeiros A. K., Song M., and Verbeek H. M. W. Business process mining: An industrial application. *Inf. Syst.*, 32(5):713–732, July 2007.
58. Van der Aalst W.M.P., Weijters T., and Maruster L. Workflow mining: discovering process models from event logs. *Knowledge and Data Engineering, IEEE Transactions on*, 16:1128–1142, Sept 2004.

# A Literatur Research

| Title | Author | URL |
|---|---|---|
| Difference and Union of Models | Marcus Alanen, Ivan Porres | http://link.springer.com/chapter/10.1007/978-3-540-45221-8_2 |
| Visual graph comparison | Andrews, K., Wohlfahrt, M., Wurzinger, G | http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5190876&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5190876 |
| Structural differences between two graphs through hierarchies | Archambault, D | http://www.cs.ubc.ca/nest/imager/tr/2009/Archambault_StructDiff_GI/majorDifferences.pdf |
| Generic tool for visualization of model differences | van den Brand, M., Protic, Z., Verhoeff, T. | http://dl.acm.org/citation.cfm?id=1826160 |
| Delta Analysis: A Hybrid Quantitative Approach for Measuring Discrepancies between Business Process Models | Esgin, E., Senkul, P. | http://link.springer.com/chapter/10.1007/978-3-642-21219-2_38 |
| Visualizing Differences between Two Large Graphs | Geyer, M., Kaufmann, M., Krug, R | http://link.springer.com/chapter/10.1007/978-3-642-18469-7_38 |
| Merging business process models. | La Rosa, M., Dumas, M., Uba, R., Dijkman, R. | http://link.springer.com/chapter/10.1007%2F978-3-642-16934-2_10 |
| Differences between versions of UML diagrams. | Ohst, D., Welle, M., Kelter, U. | http://dl.acm.org/citation.cfm?id=940102 |
| Difference computation of large models | Treude, C., Berlik, S., Wenzel, S., Kelter, U. | http://www.researchgate.net/publication/221560189_Difference_computation_of_large_models/links/00b7d525ee6c0e4c77000000 |

Table 9

# B Literatur Research extended

| Title | Author | URL |
|---|---|---|
| Business alignment: using process mining as a tool for Delta analysis and conformance testing | van den Aalst, W.M.P. | http://link.springer.com/article/10.1007/s00766-005-0001-x |

| | | |
|---|---|---|
| Delta analysis with workflow logs: aligning business process prescriptions and their reality | Kleiner, N. | http://link.springer.com/article/10.1007/s00766-005-0004-7 |
| Measuring Similarity between Business Process Models. | van Dongen, B.F., Dijkman, R., Mendling, J. | http://link.springer.com/chapter/10.1007/978-3-540-69534-9_34 |
| Matching and merging of statecharts specifications. | Nejati, S., Sabetzadeh, M., Chechik, M., Easterbrook, S., Zave, P. | http://dl.acm.org/citation.cfm?id=1248839 |
| A State-of-the-Art Survey on Software Merging | T. Mens | http://dl.acm.org/citation.cfm?id=567178 |
| Merging graph-like object structures | Albert Zündorf, Jörg P. Wadsack, and Ingo Rockel | http://www.cs.uni-paderborn.de/uploads/tx_sibibtex/SCM01.pdf |
| Meaningful change detection in structured data | Sudarshan S. Chawathe and Hector Garcia-Molina | http://dl.acm.org/citation.cfm?id=253266 |
| On a relation between graph edit distance and maximum common subgraph | Bunke, H. | http://www.sciencedirect.com/science/article/pii/S0167865597000603 |
| Graph Matching Algorithms for Business Process Model Similarity Search | Dijkman, R.M., Dumas, M., Garcia-Banuelos, L. | http://link.springer.com/chapter/10.1007/978-3-642-03848-8_5 |
| Merging Event-Driven Process Chains | Gottschalk, F., van der Aalst, W.M.P., Jansen-Vullers, M.H. | http://link.springer.com/chapter/10.1007/978-3-540-88871-0_28 |
| Detecting and resolving process model differences in the absence of a change log. | Küster, J.M., Gerth, C., Förster, A., Engels, G. | http://link.springer.com/chapter/10.1007/978-3-540-85758-7_19 |
| ASK-GraphView: A large scale graph visualization system. | J. Abello, F. van Ham, and N. Krishnan | http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4015416&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D4015416 |
| Graph visualization techniques for web clustering engines | E. Di Giacomo, W. Didimo, L. Grilli, and G. Liotta | http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4069238&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D4069238 |
| Multilevel visualization of clustered graphs. | P. Eades and Q. Feng | http://link.springer.com/chapter/10.1007/3-540-62495-3_41 |
| JGraph | | http://www.jgraph.com |

| | | |
|---|---|---|
| Using Graphviz as a Library | Emden R. Gansner | http://bsdforge.com/docs/userdocs/graphviz/pdf/libguide.pdf |
| qgraph: Network Visualizations of Relationships in Psychometric Data | Epskamp, S., Cramer, A. O., Waldorp, L. J., Schmittmann, V. D., and Borsboom, D | http://www.jstatsoft.org/v48/i04/paper |
| Perspective Oriented Business Process Visualization | Jablonski, S., and Goetz, M. | http://link.springer.com/chapter/10.1007/978-3-540-78238-4_16 |
| A System for Graph-Based Visualization of the Evolution of Software | Collberg, C., Kobourov, S., Nagra, J., Pitts, J., and Wampler, K. | http://dl.acm.org/citation.cfm?id=774844 |
| Visual Change Tracking for Business Process Models | Kabicher, S., Kriglstein, S., and Rinderle-Ma, S. | http://link.springer.com/chapter/10.1007/978-3-642-24606-7_41 |
| Weighted Graph Comparison Techniques for Brain Connectivity Analysis | Alper, B., Bach, B., Henry Riche, N., Isenberg, T., and Fekete, J. D. | http://dl.acm.org/citation.cfm?id=2470724 |
| Visualization of Search Results: A Comparative Evaluation of Text, 2D, and 3D Interfaces | Sebrechts, M. M., Cugini, J. V., Laskowski, S. J., Vasilakis, J., and Miller, M. S. | http://dl.acm.org/citation.cfm?id=312634 |

Table 10

# C Survey Screenshots

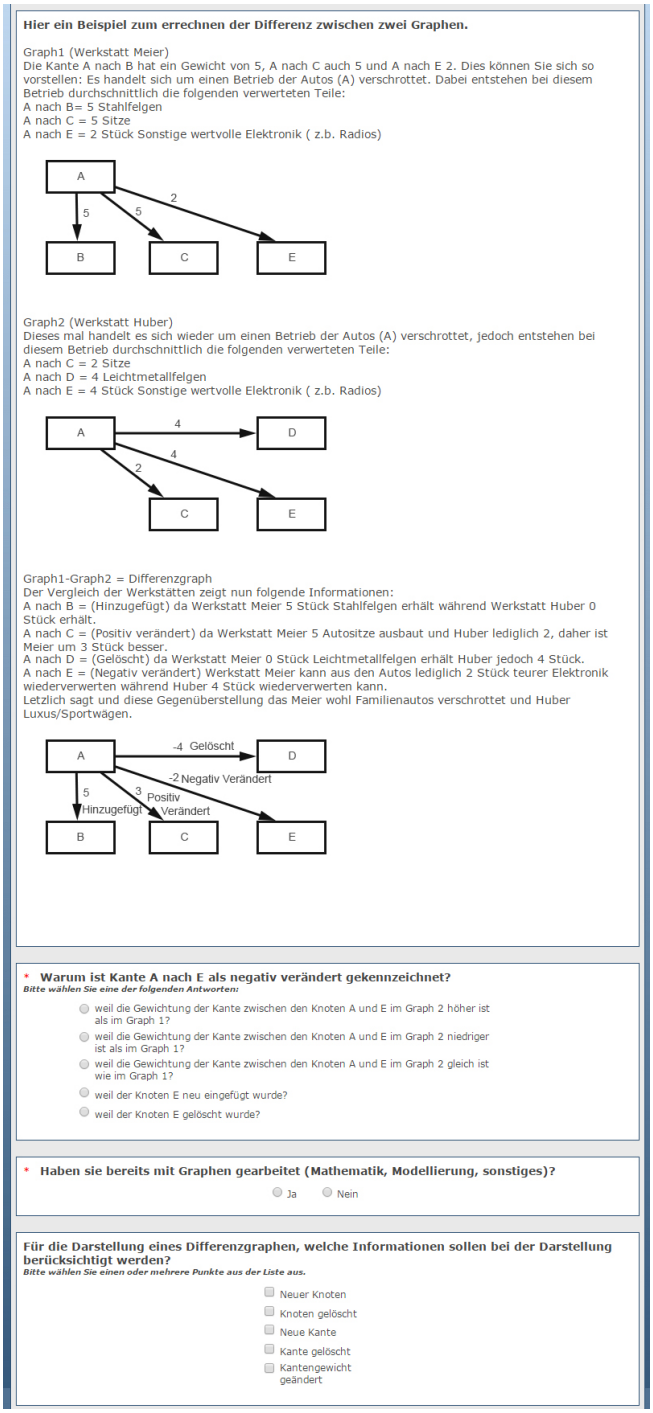



Figure 27: Introduction page

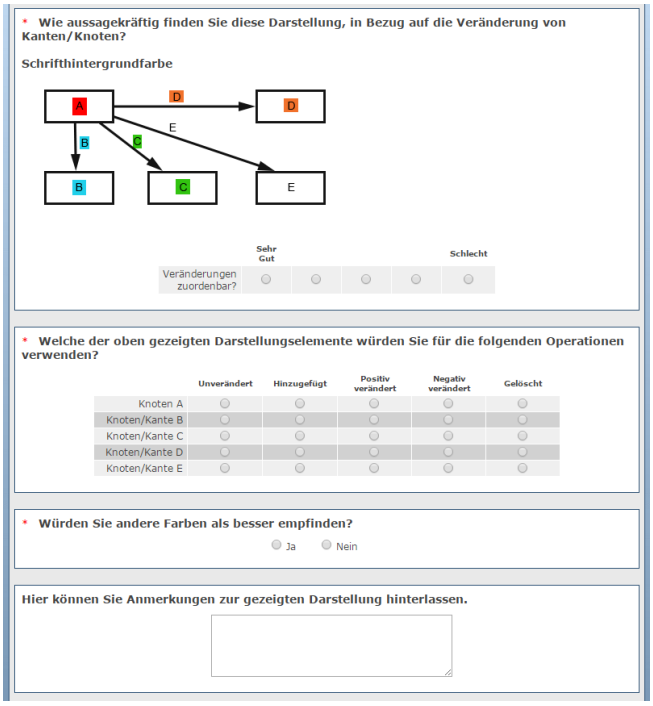



Figure 28: One of the nine representation style

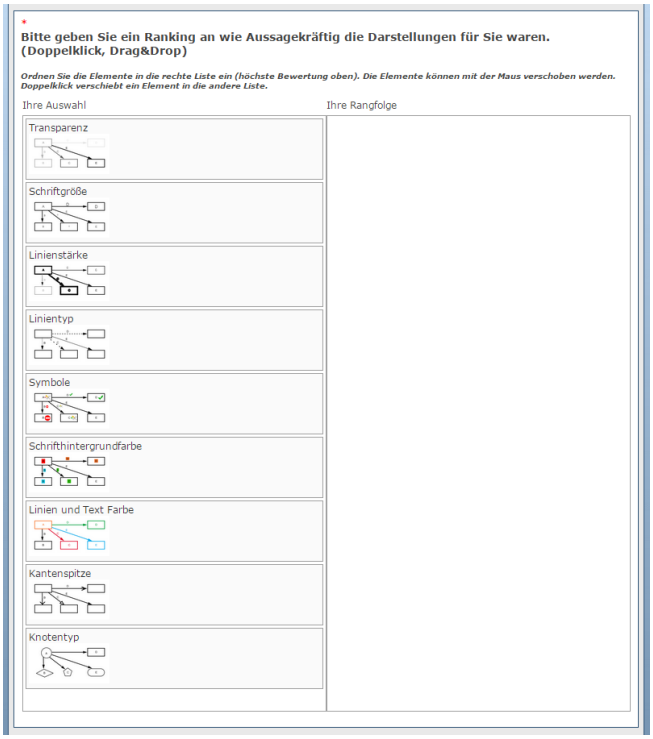



Figure 29: Ranking

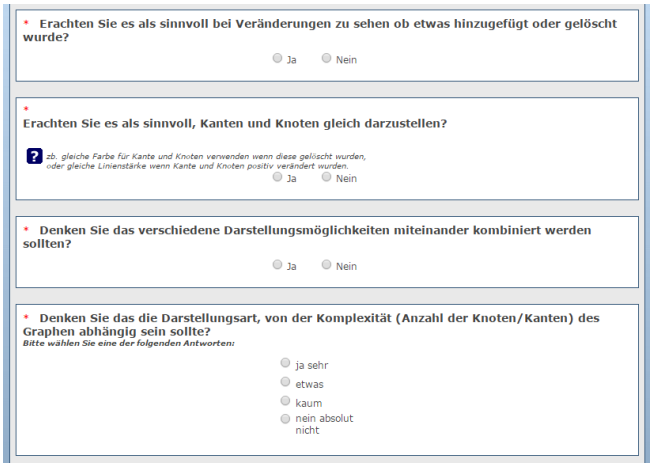



Figure 30: Additional questions

# D Abstract

## D.1 English

With the increasing usage of computer information systems more data is generated during process execution. Based on this data process mining allows analysis of the executed process. Expressing the differences between two processes is a challenging task which cannot be fulfilled by currently available tools. Based on an evaluation this thesis shows how to visualize the difference between two processes followed by an prototype implementation.

## D.2 German

Aufgrund der steigenden Nutzung von Computer Informationssystemen werden während der Prozessausführung mehr Daten den je generiert. Prozess Mining Tools nutzen diese Daten zur näheren Analyse des ausgeführten Prozesses. Mit derzeit verfügbaren Tools ist es nicht möglich die Unterschiede zweier Prozesse visuell darzustellen. Anhand einer Umfrage zeigt diese Arbeit wie die Unterschiede zweier Prozesse hervorgehoben werden können. Die Ergebnisse der Umfrage werden sich auf die Prototyp Implementierung zur Visualisierung der Unterschiede auswirken.

## E Summary - German

Das Ziel dieser Arbeit war es, die Differenz zweier Prozesse zu berechnen und die Unterschiede zu visualisieren. Eine Einführung in das Gebiet des Prozessmining sorgte für ein fundamentales Verständnis wie aus Logfiles Prozessmodelle erstellt werden.

Um die Unterschiede zweier Prozessmodelle zu visualisieren, wird auf ein Konzept zurückgegriffen, welches jedem Knoten und jeder Kante eine Markierung zuweist. Handelt es sich bei den Prozessmodellen um Modelle mit Gewichten an den Kanten (bsp. Heuristics Net), so können bei der Differenzberechnung fünf verschiedene Markierungen (Neu, Positiv verändert, Unverändert, Negativ verändert und Gelöscht) zugewiesen werden. Verfügt das Modell über keine Gewichte (bsp. Petri Net), sind lediglich drei Markierungen (Neu, Unverändert und Gelöscht) möglich.

Eine Literaturrecherche führte zu keiner geeigneten Darstellungsform um diese Markierungen zu visualisieren. Daher wurden aus der Literatur verschiedene Visualisierungsformen extrahiert. Basierend auf diesen Darstellungsformen wurde eine Umfrage, mit dem Ziel festzustellen, welche Darstellungsform bevorzugt wird, erstellt. Die farbkodierte Darstellungsform hat sich durchgesetzt und wurde für die Implementierung herangezogen.

Für die Implementierung wurden Process Mining Tools analysiert. Das Tool ProM bot die beste Grundlage, um ein Plug-in zu entwickeln. Die bereits implementierten Miningalgorithmen, Fuzzy-, Heuristics- und Alpha-miner, wurden für den Prototypen als relevant identifiziert.

Der Prototyp umfasst zwei mögliche Inputs, Logfiles und Prozessmodelle. Handelt es sich bei dem Input um Logfiles wird ein Menü geöffnet, welches die Selektion auf einen der oben angeführten Miningalgorithmus erlaubt. Bei einem Prozessmodell als Input werden die Typen Heuristics-, Petri- und Activity-Netz erlaubt. Beide Möglichkeiten, das Tool zu starten, führen dazu, dass der Input in ein Activity-Netz transformiert wird. Basierend auf dem Activity-Netz erfolgt die Berechnung der Differenz, welche in einem neuen Modell gespeichert wird. Jeder Kante und jedem Knoten dieses Modells werden während der Berechnung Markierungen zugewiesen. Als letzter Schritt erfolgt die Visualisierung des Differenz-Modells, bei welchem die Kanten und Knoten in verschiedenen Farben dargestellt werden. Die darzustellenden Farben basieren auf den zugewiesenen Markierungen.

Der implementierte Prototyp zeigt das sich Konzept, Anforderungen und Evaluierungsergebnisse vereinen lassen.

# F Curriculum Vitae

| **Persönliche Daten** | |
|---|---|
| Name: | Manuel Gall |
| **Praxiserfahrung** | |
| Februar 2013 - September 2014 | Programmieren & Marketing bei Together Internet Service GmbH |
| **Ausbildung** | |
| ab Oktober 2012 | Master Wirtschaftsinformatik |
| Oktober 2009 - Juli 2012 | Bachelorstudium Informatik mit Schwerpunkt Wirtschaftsinformatik |