Mining document-centric process models
the product data model case

van Welie, M.

Award date:
2012

Link to publication

Mining Document-centric Process Models

The Product Data Model case

Master Thesis

In Partial Fulfillment of the
Requirements for the Degree of

Master of Science in Business Information Systems

By

Michel van Welie

Supervisors:

dr. M. Comuzzi (Marco)	**TU/e**
dr.ir. I.T.P. Vanderfeesten (Irene)	**TU/e**
dr. C. Stahl (Christian)	**TU/e**
J. Klok (Jakob)	**Océ Technologies**

Eindhoven, March 2012

Abstract

Business Process Modeling (BPM) is used in many businesses to analyze and improve all sorts of business processes. Traditionally the focus of BPM has been on the control-flow of a process. More and more businesses store information about their daily processes in information systems, making this information available to automatically generate process models based on this available data. One of the techniques used to derive a process model from data stored in an information system is process mining. Process mining provides a means to (re)discover and analyze business processes. Process mining is mainly used to discover and analyze the control-flow perspective of a process.

In Artifact-based Business Process Modeling however, the focus is on the data that is used during a process in order to get to a certain product or goal. There are several ways to model artifact-based or artifact-centric processes; however, there are no process mining techniques or other ways available to automatically derive process models from such processes that focus on the data that is used during these processes.

This project describes how to apply process mining techniques to a document-centric environment in order to generate process models with a focus on data in an automatic way. Furthermore, a plug-in, called the PDM Miner plug-in, is created for the ProM framework. This plug-in is then applied to real data from a case management system called Serenga (formerly known as DossierFlow), developed by Océ Technologies. The feedback that we got from both experts and practitioners is discussed and suggestions are made for future work and improvements that can be made.

Contents

1 Introduction

More and more businesses store information about their daily processes in information systems and more often this information is used to monitor and improve processes within these businesses. Process mining is therefore becoming an import way to analyze (business) process information. It can give useful insight into how processes are executed and where possible improvements can be made. In most cases however process mining focuses on the control-flow aspect of processes, i.e. the order in which certain activities or steps in a process are executed. This report presents a way to automatically generate a model, with the focus on the data that is used during a process rather than control-flow of a process, using process mining techniques on available usage data. This is done by adapting and applying existing process mining techniques to a document-centric situation to extract a generic model from a document log. The solution provided in this report is developed for and tested on a case management system, called Serenga, developed by Océ.

1.1 Context

When analyzing (business) processes, traditionally the focus is on the control-flow of such processes. For some processes however, for example the processes supported by case-handling systems, a data-driven approach will, in general, give a better and much easier to understand representation of the processes than conventional (control-flow oriented) workflow/process techniques.

This project is done in collaboration with the Research and Development (R&D) department of Océ Technologies B.V. Océ, founded in 1877 and based in the Netherlands, is one of the leading companies providing printing and document-management services throughout the world. In 2010 a collaboration between Océ and Canon was made official and Océ became part of the Canon Group, making it the world's biggest supplier of printers.

Serenga (formerly known as DossierFlow) is a web-based case management system developed by Océ, which allows customers to digitally keep track of their cases ('dossiers' in Dutch) and which helps the users with their case-handling processes. Usage data from some Serenga customers is available and can be used for this project.

1.2 Problem Description

Up until now creation of an artifact or data-centric process model has been a manual task, mainly focused on the design of a process. Manual creation of these models is a very time consuming task, taking days or weeks to get a good understanding of the domain and taking interviews to find out how people perform their daily tasks and what data is used during these tasks. Apart from the time this manual process takes, it might also lead to a process that is correct according to the way people *think* the process is or should be executed and what data is necessary to complete a process, but in reality this might be (slightly) different. Therefore, there is a need to turn this manual process into an automated process using historical usage data that is available on the way processes have been executed in the past.

For Serenga there is not a strict, pre-defined way of how users should use the system. Because of this, the processes that Serenga is used for in most cases do not have a strictly defined structure. Instead of restricting the user by limiting possible behavior to force a more structured way of working, providing the users with some kind of visualization of a process helping the user on how to perform a certain process would make these processes more structured without limiting the user in his/her behavior.

1.3 Project Goals

The goal for this project with respect to Serenga is to generate models which represent the way the users use the system for their everyday processes. These models should be based on historical usage data that is provided/stored by Serenga. With these models, Océ will be able to provide better/additional services to their customers. The customers of Serenga can use these models of their processes to get insights into how these processes are actually being executed, so the models should be easy to understand and should not contain too much (irrelevant) information. Furthermore, these models can be used to guide the user of Serenga through a process by giving the user the possibility to see how this process was executed in earlier cases. Therefore, these models should contain information on how processes were executed in the past to help the user to decide what possible next steps can be taken in order to finish the current process.

The idea is to use process mining techniques to generate a process model in a data-driven environment (in this case the data will be documents) in order to reduce the effort that is needed to get to such a model. Moreover, these models should give a better representation of the underlying process in contrast to manually created process models.

1.4 Approach

The approach taken for this project is to first research the different available possibilities which could be used to model the behavior in a document-centric environment. This is done in combination with an investigation into different process mining techniques to see which combination is best suited to apply to the given situation of Serenga.

After the decision is made which methodology will be used and which process mining technique to use, the chosen mining technique is altered in such a way that it can be used in combination with the selected methodology and that it can be used for data instead of control-flow.

For this project, the decision is made to adapt the Flexible Heuristics Miner (FHM) algorithm. The FHM is chosen because it works well on unstructured data. Since Serenga provides a rather ‘free’ environment, this results in unstructured processes and hence the FHM should be able to perform well in this situation. The adapted algorithm will take a document log file, containing multiple traces with events that represent which documents where used during a process, after which a Product Data Model (PDM) will be derived from this log. As a result, this will provide an automated way to generate document-centric process models in a data-driven approach. The algorithm is implemented in JAVA and is made as a plug-in for the process mining framework ProM (version 6.1).

The last step is to apply and test the adapted mining technique with historical Serenga usage data to generate models that correctly represent the actual real life processes of which information is stored by Serenga and to create a way to make the results of the mining process available so they can be accessed from within Serenga so that these results can be used by the users of the system. This is done by creating a webpage that shows the PDMs corresponding to the process that is being executed.

1.5 Framework

Figure 1 shows the conceptual framework that is used to get from data stored by a document-centric application to a process model. The framework shows the different steps that need to be taken in order to generate these process models. The first step is to create a document log based on the data that is stored by the application. After this log has been created process mining techniques can be used to discover a process model. For document-centric processes (and data-centric processes in general) however, no process mining techniques are currently available. That is why, in this project, we adapt current process mining techniques designed for processes focused on control-flow to this document-centric setting. The last step is to do some evaluation of the generated models.

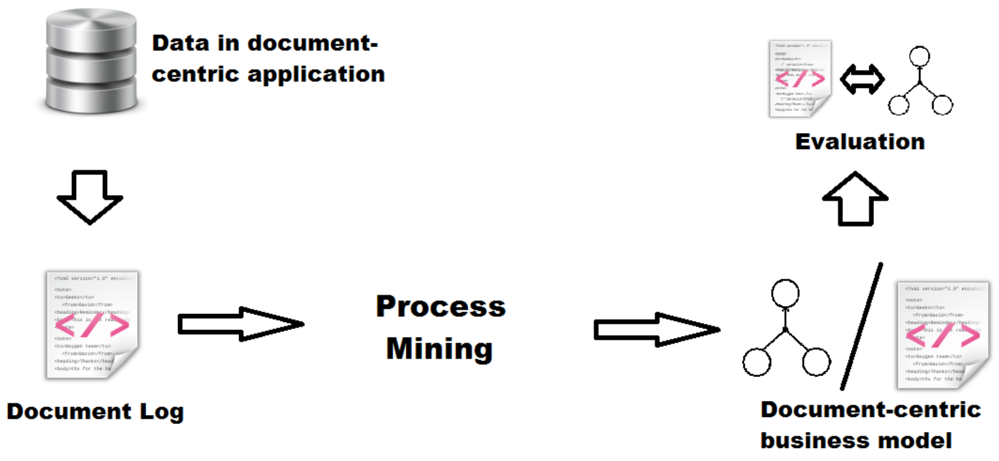

Data in document-centric application
Evaluation
Document Log
Process Mining
Document-centric business model

Figure 1 Conceptual framework

For this particular project we used data available from an application called Serenga in order to create the document log. We have chosen to use the Product-Based Workflow Design methodology to model the processes (that is, the relation between different documents) and we have created a process mining plug-in called the PDM Miner to mine a document log to get a Product Data Model (PDM) that represents the process. This results in the following overview of this project:

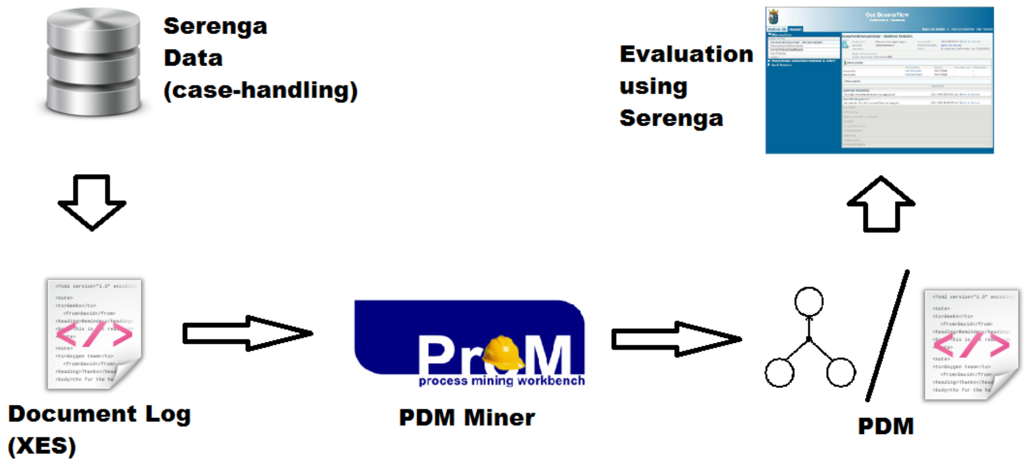


Figure 2 Project framework

1.6 Overview

This section gives an overview of the content of the coming chapters.

Chapter 2 discusses relevant background information. The first section discusses some related work on different artifact-based and data-centric workflow design methods and different process mining techniques that are relevant to the given situation. The second part of the chapter describes preliminary information that is required to understand the remainder of this report

Chapter 3 introduces a running example that is used throughout the rest of the report to explain the different steps that are taken.

Chapter 4 introduces the document log and what data should be available in order to create such a log.

Chapter 5 contains the main contribution of this project and explains the algorithm that is used to turn a document log into a PDM.

Chapter 6 presents the results of applying the algorithm to usage data of Serenga and shows different models generated by the algorithm.

Chapter 7 discusses some points that came up during a workshop session with experts on PBWD and practitioners working on Serenga.

Chapter 8 shows the conclusions that follow from this project and also proposes some future work that could be done to improve the algorithm and the generated models.

2 Background

This chapter describes background information that is relevant for this project. Section 2.1 provides an overview of some related work that has been done with respect to Artifact-centric Business Process Modeling and Process Mining. Section 2.2 gives preliminary information that is required in order to be able to understand the remainder of this report.

2.1 Related Work

This section is split into two parts. The first part (section 2.1.1) describes related work on data-centric or Artifact-centric Business Process Modeling approaches. In contrast to the traditional process- or activity-centric modeling that focuses on the control-flow of a process, artifact-centric process modeling focuses on the data that is used during the execution of a (business) process instead of focusing on the activities and the control-flow of the process. The second part of this chapter (section 2.1.2) focuses on related work on process mining and process mining techniques.

2.1.1 Artifact-centric Business Process Modeling

In Artifact-centric Business Process Modeling [1],[2] the main focus is on the product of a process and the changes or updates made to business data by some actions or operations in the process which lead to this (final) product. The traditional way of looking at business processes has been to look at the activities within a process, i.e. what actions are taken to get to a certain goal [3]. The artifact-centric approach is a more data-driven approach in comparison to the traditional situation where the focus is on the control-flow of a process. This section describes different techniques and approaches related to artifact-centric modeling.

2.1.1.1 Business Artifacts

Nigam and Caswell [1] first introduced the concept of *artifacts* in the context of a technique called OpS (Operational Specification), for constructing formal operation descriptions of a business. They describe an artifact as:

"*a concrete, identifiable, self-describing chunk of information that can be used by a business person to actually run a business*".

Each artifact has its own lifecycle, during which its contents can be changed by *business tasks* and during which an artifact can be stored in a *repository*. These business tasks and repositories are connected through "*transport pipes*" through which artifacts and artifact content can be exchanged. When a task receives an artifact, only the information present in the artifact at that time can be used by that task. Interaction of two lifecycles can occur in two ways, either a reference to another artifact is made, or another artifact needs to be modified. In the latter case, there exists a task that is part of both lifecycles. In [2] a formal model is introduced for this kind of artifact-centric business processes.

Using operational models it becomes possible to describe the operation (or processes) of a business in one single representation, where information is represented by artifacts and activities by tasks. However, the creation of such models is a manual task. Figure 3 shows an OpS diagram for a grocery shopping business taken from [1] in which the artifacts, tasks and repositories are shown.

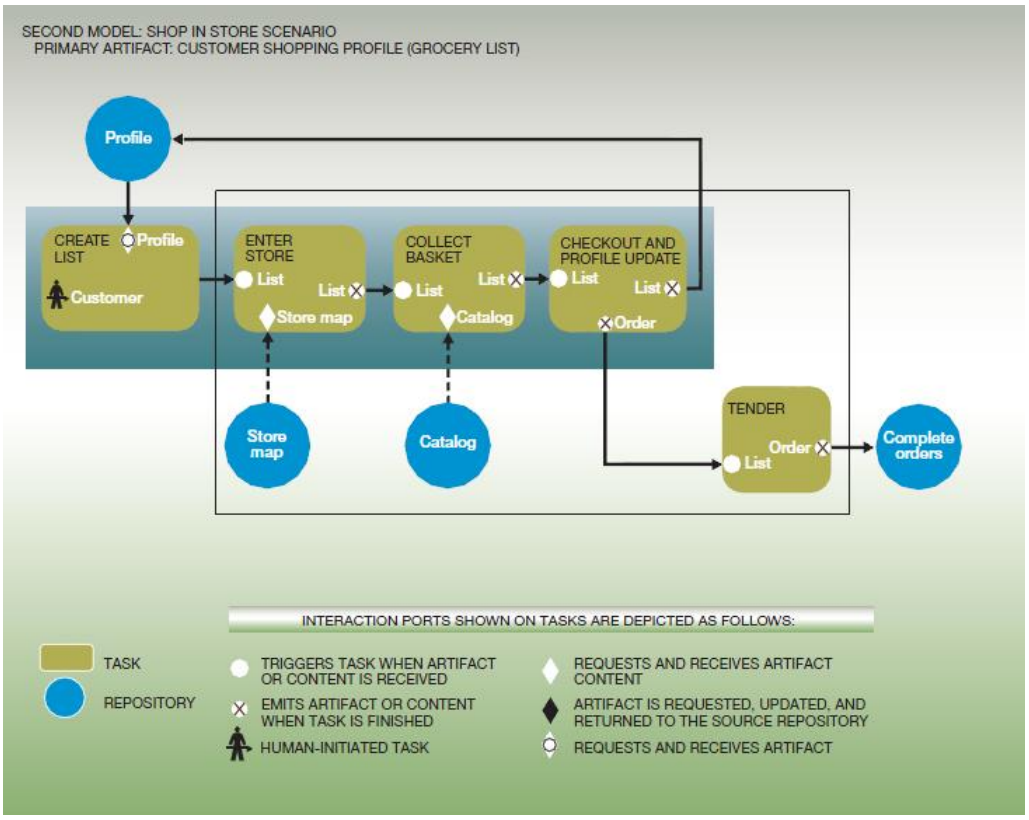

SECOND MODEL: SHOP IN STORE SCENARIO
PRIMARY ARTIFACT: CUSTOMER SHOPPING PROFILE (GROCERY LIST)
Profile
CREATE LIST
Profile
Customer
ENTER STORE
List
List
Store map
COLLECT BASKET
List
List
Catalog
CHECKOUT AND PROFILE UPDATE
List
List
Order
Store map
Catalog
TENDER
Order
List
Complete orders
INTERACTION PORTS SHOWN ON TASKS ARE DEPICTED AS FOLLOWS:
TASK
REPOSITORY
TRIGGERS TASK WHEN ARTIFACT OR CONTENT IS RECEIVED
EMITS ARTIFACT OR CONTENT WHEN TASK IS FINISHED
HUMAN-INITIATED TASK
REQUESTS AND RECEIVES ARTIFACT CONTENT
ARTIFACT IS REQUESTED, UPDATED, AND RETURNED TO THE SOURCE REPOSITORY
REQUESTS AND RECEIVES ARTIFACT

Figure 3 Model of grocery shopping business

2.1.1.2 *Document-driven Workflow Modeling*

In Document-Driven Workflow additional properties are added to documents. In [4] and [5] different features are stated. Traditional metadata of a document is extended with other information that might be relevant for a user or program, creating a uniform interface for all document interactions. There are also user-specific properties which are related to individual users of a document. Active properties are properties which can be used to control document behavior. The code within these active properties can be executed, which can be used to control or augment document functionality, by for example extending the operations on a document and the way a document interacts with other documents.

In [4] a new approach called "*document-centered collaboration*" is presented to integrate content and coordination functionality into documents. This is done by creating a middleware solution between applications and document repositories. In [5] the focus is on the issues that arise when adding these active properties into document management systems. In [6] a framework for document-driven workflow systems is given in which documents consist of DocElements which in turn consist of a number of DataFields. A document can receive an event when an element or data field is changed, or it can receive events from other documents. Hence, the process is driven by (changes to) documents.

2.1.1.3 *Product-Based Workflow Design*

Product-Based Workflow Design (PBWD) is a methodology to design and analyze business processes by looking at the product of a process [7]. In PBWD a Product Data Model (PDM) is used as the underlying model to describe the structure of a process. A PBWD project consists of four phases. In the first phase, the scoping phase, the process which is to be (re)designed is selected and the product of this process is identified. In the analysis phase, the product is analyzed and its data elements and operations on these data elements are determined. These data elements and operations are put in a PDM that represents the structure of the selected process. The third phase is the design phase, in which several process models are derived from the PDM that was created in the previous phase. In the last phase, the evaluation phase, the different process models are evaluated and the best one is selected.

Furthermore, a way of directly executing a PDM, instead of first creating process models, is also described in [7]. This approach is called Product Based Workflow Support (PBWS). Instead of creating Petri nets that can be executed, this step is skipped and the model is directly executed using Marcov Decision Processes (MDP).

PBWD has already been applied in the process mining field and some plug-ins concerning PDMs are already available for ProM (version 5.2), including importing, visualization and analysis plug-ins.

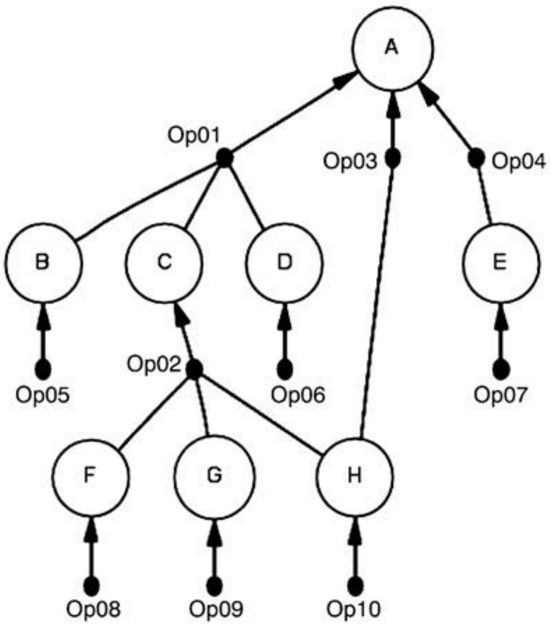
Product Data Model (PDM)

A Product Data Model (PDM) describes a process by looking at the data elements that are used/changed during this process to get to a certain goal or product. A PDM consists of a set of *data elements* and a set of *operations* on these data elements (similar to the artifacts and tasks from section 2.1.1.1). An operation also contains a (set of) condition(s), a set of resources and might contain a number of non-functional properties that are relevant for a process.

An operation has zero or more data elements as its input and exactly one data element as its output. Also, an operation has conditions regarding its input data elements. An operation is enabled when all conditions regarding its input elements have been satisfied. Each operation has a resource (class) assigned to it. Operations can also have a number of non-functional properties.

A PDM has one special data element which is called the *root*. This root element represents the final decision (or goal) in a process. Executing an operation that has the *root* element as its output marks the end of the process.

Figure 4 shows an example PDM (created with an earlier version of the ProM framework) of a process that determines a maximum mortgage taken from [8].



A. Maximum mortgage
B. Percentage of interest
C. Annual budget to be spent on the mortgage
D. Term of mortgage
E. Previous offer (within the period of validity of the offer)
F. Percentage of income to be spent on the mortgage
G. Gross income per year
H. Credit registration

Figure 4 Example PDM [8]

2.1.1.4 Case-handling
In [9],[10] it is stated that in a lot of situations, workflow management systems are not flexible enough because the focus is on control flow and the routing focuses on what should be done, instead of what can be done. This is where case handling comes in. Case handling offers a combination between data-centric and control-flow approaches and hence is more flexible, assisting the user what can be done instead of what should be done.

In case handling the main concept is a case, which is routed by activities. The state of a case however is not specified by the control-flow, but by the status of its data objects. Each activity can be executed by a certain role, which is a group of (case-) workers (or actors). This role determines which data will be available for the person executing the activity. The use of roles allows for the distribution of work to a group of workers and the workers decide which step needs to be taken next based on the information that is available on a certain case. Hence, instead of letting the system choose which steps to take during a process, possible next steps are provided to the worker of the system based on the availability and status of data elements so that the case worker can then decide on what steps to take next.

For each activities and process three roles are specified: The execute role is required to start a process or carry out an activity. The redo role used to undo activities so they can be carried out again. The skip role allows skipping activities. Together, the states of the activities and the states of the data objects define the state that a case is in. Whether or not an activity can be executed in a certain part of the process is defined by preconditions that depend on (the state of) data objects. The same goes for post-conditions.

2.1.1.5 Research gap in Artifact-centric Business Process Modeling
As shown in the previous section, there are many ways to model artifact-centric business processes. However, for all these modeling techniques the information that is needed to construct these models has to be obtained manually. The way this information is obtained is by looking at available data and

interviewing the people that are involved in the execution of the process that is being modeled. Since all of this is done manually, this is a very time consuming process taking weeks or even months to get a good overview of how a process is being executed. Table 1 taken from [11] sums up the pros and cons of a number of different data-gathering techniques.

Technique	Good for	Kind of data	Pros	Cons
Questionnaires	Answering specific questions	Quantitative and qualitative data	Can reach many people with low resource.	The design is crucial. Response rate may be low. Responses may not be what you want.
Interviews	Exploring issues	Some quantitative but mostly qualitative data	Interviewer can guide interviewee. Encourages contact between developers and users.	Time consuming. Artificial environment may intimidate interviewee.
Focus groups and workshops	Collecting multiple viewpoints	Some quantitative but mostly qualitative data	Highlights areas for consensus and conflict. Encourages contact between developers and users.	Possibility of dominant characters.
Naturalistic observations	Understanding context of user activity	Qualitative	Observing actual work gives insights that other techniques cannot give.	Very time consuming. Huge amounts of data.
Studying documentation	Learning about procedures, regulations, and standards	Quantitative	No time commitment from users required	Day-to-day work will differ from documented procedures.

Table 1 Data-gathering techniques [11]

As this table shows, most of these techniques might take a considerable amount of time. Furthermore, models that are the result of such manual approach might not necessarily reflect the actual way a process is executed. The models will give a good representation of how people *think* a process is or should be executed, but this might differ from how a process is actually executed in reality.

For traditional activity-based processes, this process can be automated by using process mining techniques to retrieve process models from data stored in logs/databases. This can save a lot of time, if the right data is available. Another benefit of using an automated way to generate models using process mining techniques is that real usage data is used that better represents the process that was executed. This as opposed to process models created using manual data gathering techniques shown above, which might differ from the execution in reality.

For the reasons mentioned above, we want to apply process mining techniques to artifact-centric business processes (in our case the artifacts will be documents). Since no literature is available yet on this subject, this project will be a first step towards automating the creation of process models in an artifact-centric setting. We choose to use PBWD as an illustrating example to show how to apply process mining techniques to a document-centric approach.

2.1.2 Process Mining

Process mining is a way to retrieve and analyze useful information from event logs which are retrieved from information systems. As more and more businesses store information about there every day processes in information systems, the use of process mining becomes more and more interesting. With process mining, the (historical) information stored in these information systems can be used to give insights into how a business' processes are actually being executed which in turn can be helpful to improve these processes.

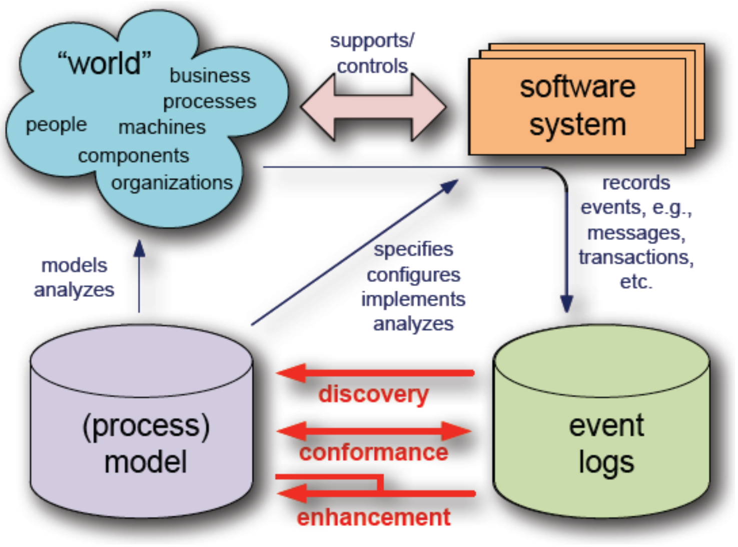


Figure 5 Three types of process mining [12]

Process mining techniques can be divided into three classes: Discovery, Conformance and Extension [12]. Discovery techniques are used to create (process) models that represent the information stored in information systems about a certain process. Conformance techniques are used to check to what degree a model actually resembles the behavior stored in an information system and vice versa. Extension (or enhancement) techniques use an already existing model in combination with data in order to improve or extend the existing model.

2.1.2.1 ProM framework

To create an environment to implement process mining techniques, several people of the Eindhoven University of Technology created a generic open-source framework called ProM[1]. ProM is one of the most well-known process mining tools and is widely used, especially in scientific environments. The ProM framework supports various process mining techniques and algorithms by means of plug-ins. Since its creation, a lot of plug-ins have been developed and new plug-ins are being developed and added regularly. With the new version of ProM however, not all plug-ins available in previous versions (ProM 5.2) have been available/ported (yet). For example, in ProM 5.2 there existed several

[1] The ProM Framework is available at: www.processmining.org

plug-ins related to the Product Data Model mentioned before, which are not yet available in ProM 6. The mining plug-ins that are mentioned in this report however, have all already been implemented in ProM 6. Besides ProM there are several other (commercial) process mining tools available.

A first step to apply process mining to data-centric processes was made by Günther and van der Aalst [13], who applied the multi-phase mining algorithm [14] to create a process model (Petri net) of a log of a case handling process created with FLOWer[2].

As stated before, there is no automated way to derive PDMs from available data. Hence, there are no process mining techniques available for this either. There are however a lot of plug-ins and mining algorithms that focus on the control-flow of a process. The next sections describe a few (discovery) algorithms/plug-ins that have been considered when looking for a solution for a data-centric approach taking into account the unstructured nature of the processes.

2.1.2.2 Alpha Miner

The first process mining algorithm created was the α-algorithm [15]. The algorithm itself only consists of 8 rules and is therefore a very small and simple algorithm. Yet, from a scientific point of view, the value of the algorithm is very significant and other algorithms have build on the principles of this algorithm.

The basic idea of the α-algorithm is that given an event log, one can go through the traces to see in what order events occurred. Several ordering relations are used to describe in what order events occurred in the log and also to determine what kind of relations (splits/joins) occur. This information is then transformed into a Workflow-net (Petri net). An example of such net is shown in Figure 6.

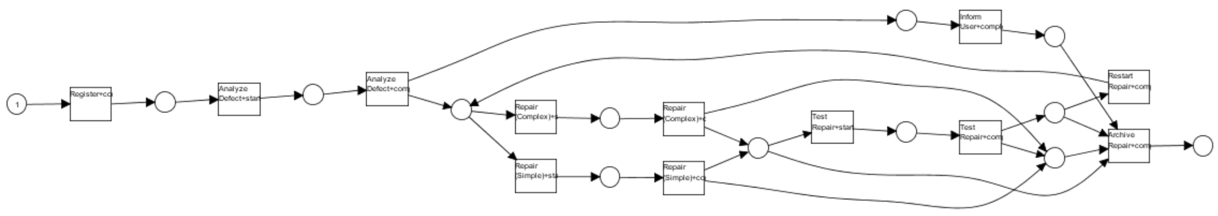


Figure 6 Example result of the α-miner plug-in

As mentioned in [16] one of the limitations of the α-algorithm is that the algorithm does not work very well with unstructured data. That is, although it creates correct models, these models may be difficult to comprehend because they contain too much information, which results in so called "spaghetti" models (see Figure 7). This is due to the fact that the α-algorithm tries to model all the behavior that can be extracted from the event log.

[2] FLOWer (now known as BPMone) is workflow management software created by Pallas Athena which was acquired by Lexmark in 2011 and became part of Perceptive Software: www.pallas-athena.com

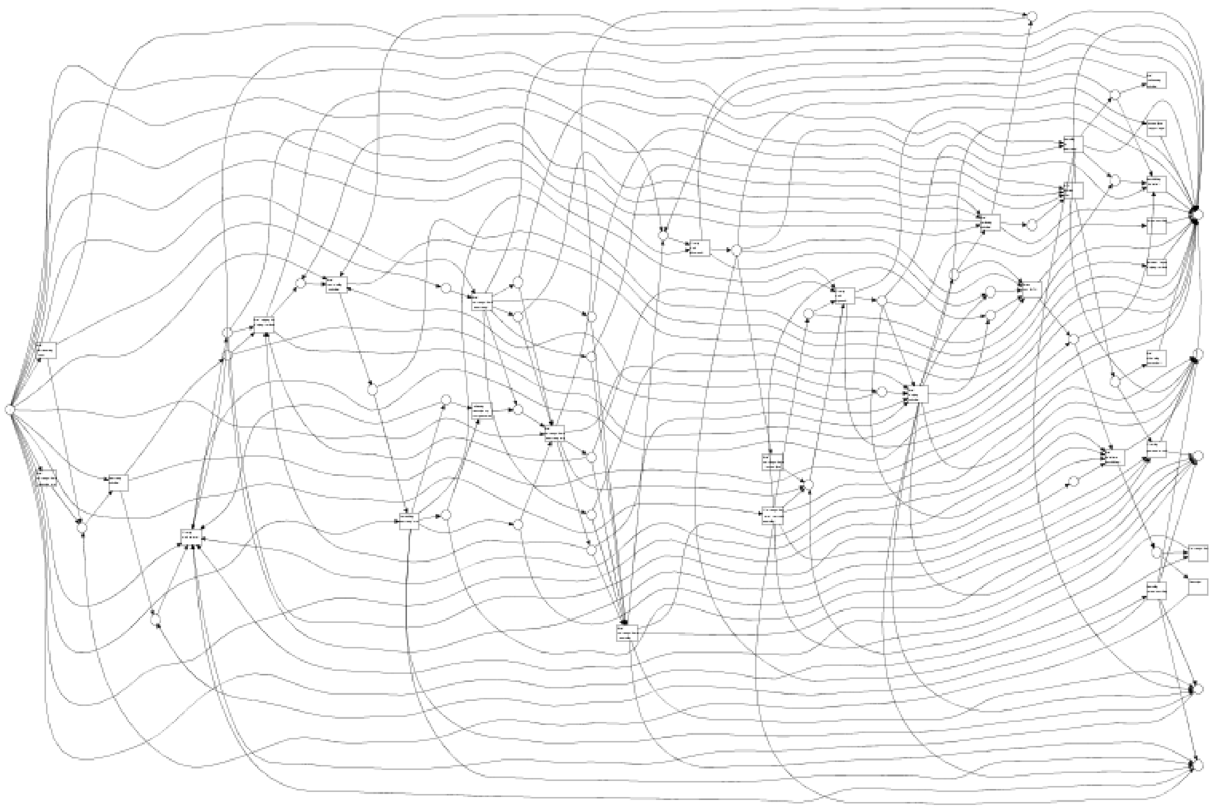


Figure 7 Example of a ‘spaghetti’ model

For this project, the data that is available is from highly unstructured processes. The way Serenga is designed allows for much flexibility when using the software, which inevitably results in unstructured processes. The use of the α-algorithm would not be preferred in such situations as this would result in models that are too complex for the customers/users of Serenga.

2.1.2.3 (Flexible) Heuristics Miner

A second process mining algorithm worth mentioning is the HeuristicsMiner algorithm [17]. The HeuristicsMiner algorithm was the next process mining algorithm developed shortly after the α-algorithm. This algorithm has some similarities with the alpha miner, but is a bit more complex. In contrast to the α-algorithm, the HeuristicsMiner does not try to model all the behavior that is present in an event log, but instead it tries to filter out non-relevant behavior (noise and uncommon/exceptional behavior) from the log to get a model that has a good clear structure and yet has all (or most of) the required detail. Figure 8 shows an example output from the HeuristicsMiner, a Heuristics net.

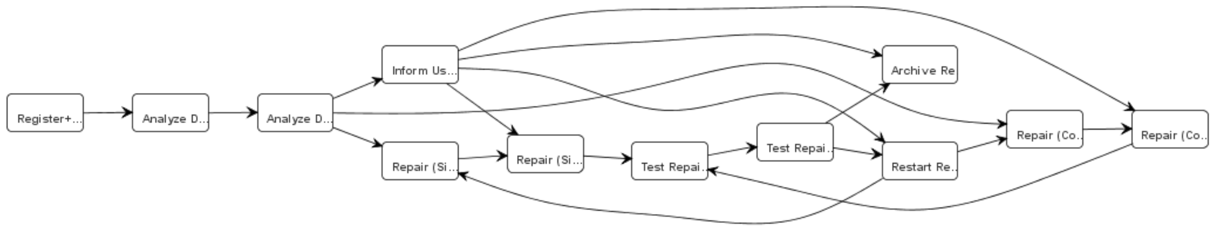


Figure 8 Example Heuristics Net

The way the HeuristicsMiner algorithm filters behavior is by using dependency measures between activities, which are based on the order in which activities appear in the event log. The more two activities (directly) follow one another, the stronger the dependency relation between these two activities is. The HeurisicsMiner has an *all-activities-connected* heuristic, which selects the best relation for each of the activities; hence making sure that all activities are included in the model. Different thresholds are used to indicate which (additional) dependency relations are accepted. This way uncommon behavior can be separated from the main behavior in a process. After this the routing information (AND/XOR splits/joins) is derived, by determining whether elements mostly occur together in one trace, or mostly in separate traces. The last step in the HeuristicsMiner algorithm is to determine long distance dependencies, which are dependencies between activities that do not *directly* follow one another, but do eventually follow one another in most of the traces. Using a good combination of threshold values results in a correct representation of the event log, where even low frequent behavior can be captured.

The Flexible Heuristics Miner (FHM) [18] is an updated version of the HeuristicsMiner. It uses (augmented) causal nets (or C-nets) as the underlying process representation language that is used to construct a model. The idea behind the Flexible Heuristics Miner is similar to that of the HeuristicsMiner. However, the implementation is a little different since C-nets are used. The FHM algorithm makes use of the same thresholds, and the *all-activities-connected* heuristic has been renamed to *all-tasks-connected*.

The algorithm that is described in this report is based on the Flexible Heuristics Miner because the algorithm used in the FHM fits best to the situation at hand. It allows for the creation of models that can be tweaked using the different thresholds so that only the most frequent behavior is contained in these models. This results in models that should be easy to understand and hence, can be used to provide the users of Serenga with models which could help in completing processes. With some minor changes the FHM algorithm will be used to output a PDM as a result of the mining process.

A downside of this plug-in however, is that the values for the different thresholds that are used may differ for every process. Hence, for each process the user would need to find the correct settings for the different thresholds to get the optimal (or at least a decent) solution.

2.1.2.4 Fuzzy Miner

The Fuzzy Miner [19] is related to the HeuristicMiner. Like the HeuristicsMiner, the Fuzzy Miner also uses heuristics to remove less relevant information from a model. Instead of simply removing elements, the Fuzzy Miner also applies clustering techniques to group certain activities. The Fuzzy Miner is designed to work well with complex and less-structured processes. As mentioned before, when modeling these processes one usually ends up with a “spaghetti” model, a model which accurately describes reality, but a big part of it represents uncommon behavior which is not really relevant or important.

The grouping of activities that ‘belong’ together reduces the number of elements in a model, resulting in a more comprehensible model. Two measures are defined, significance and correlation, to determine what behavior is preserved, aggregated or abstracted from. In contrast to the other algorithms mentioned in this chapter, the Fuzzy Miner does not extract split/join information, but simply represents relationships between (groups of) events. In the first step of the algorithm all events are translated to nodes, which get a directed connection between them if they have a relation to one another. After this, the model is simplified using three transformation steps: *conflict resolution*, *edge filtering* and *aggregation and abstraction*. A model created by the Fuzzy Miner plug-in is shown in Figure 9. The blue octagonal nodes represent clusters containing multiple activities, which are less relevant.

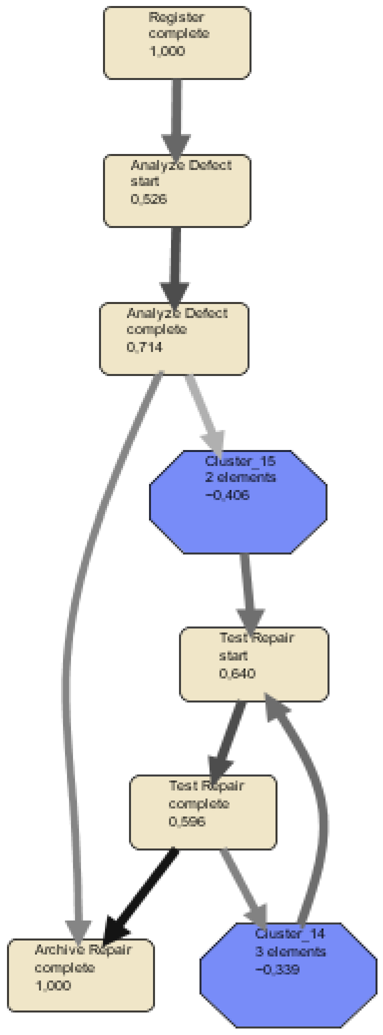


Figure 9 Fuzzy miner

In the case of Serenga however, the Fuzzy Miner would preferably not be used as the main algorithm to generate models as it does not include split information, which is important when these models have to be used as a template to guide the execution of a process. The idea of clustering or grouping of activities could however be applied in combination with a PDM. In case there are too many elements present in a PDM making the model to complex, grouping certain elements to get a higher level of abstraction could be useful.

As can be seen in Figure 9 arrows can have different shades of gray. The darker the shade of the arrow, the greater the significance of the relation it represents. This is something that we will also use in the PDMs created for this project to visualize how often a certain operation was executed.

2.2 Preliminaries

This section gives information that is needed to understand the remainder of this report as it will build upon these concepts. Two main concepts will be discussed here. Section 2.2.1 explains the concept of a Product Data Model (PDM). Section 2.2.2 introduces the event log as it is supported by the ProM framework.

2.2.1 Product Data Model (PDM)

The Product Data Model (PDM) was described in section 2.1.1.3 as part of the PBWD methodology.

A Product Data Model is a collection of data elements. Operations indicate how one data element can lead to the creation (of a value for) another data element. An operation can have zero or more input elements and one output element.

The formal definition of a PDM, based on [20] and [7], is given below.

Definition 1 (Product Data Model): *A Product Data Model (PDM) is a tuple (D, O, C, W, root, cond, res).*

- *D: Set of data elements*
- *O: Set of operations;* $O \subseteq D \times \mathcal{P}(D)$
- *C: Set of conditions; Boolean expressions*
- *W: Set of resources classes*
- $root \in D$

Each operation has the following properties:

- $cond: O \rightarrow C$*, assigns a condition to an operation*
- $res: O \rightarrow W$*, assigns a resource class to an operation*

One thing that will be important for this project is the fact that a PDM is a directed, acyclic, forward hyper graph and hence does not allow loops. This is due to the following two assumptions [7];

Assumption 2.1: *a data element can only occur once in a PDM*

Assumption 2.2: *once the value of a data element is determined, it never changes.*

Assumption 2.1 is to prevent different instances of the same element. This would otherwise create confusing models where one element can possibly have multiple values in different parts of the model. Assumption 2.2 implies that there can be no loops in a PDM. Once a data element has received a certain value, this value will remain the same throughout the (rest of the) process. For this project we will also stick to these assumptions. We need these assumptions later on when evaluating the models that are created for this project.

Routing

As mentioned, in a PDM routing is done by operations. If an element can be created in more than one way, this is represented by multiple incoming arcs (Figure 10b) and hence this represents alternative paths and is comparable to a XOR join. If multiple elements are needed to create (a value for) another element, an operation has multiple input elements (Figure 10c). This is comparable to a parallel execution (or synchronization pattern) and can therefore be seen as an AND join.

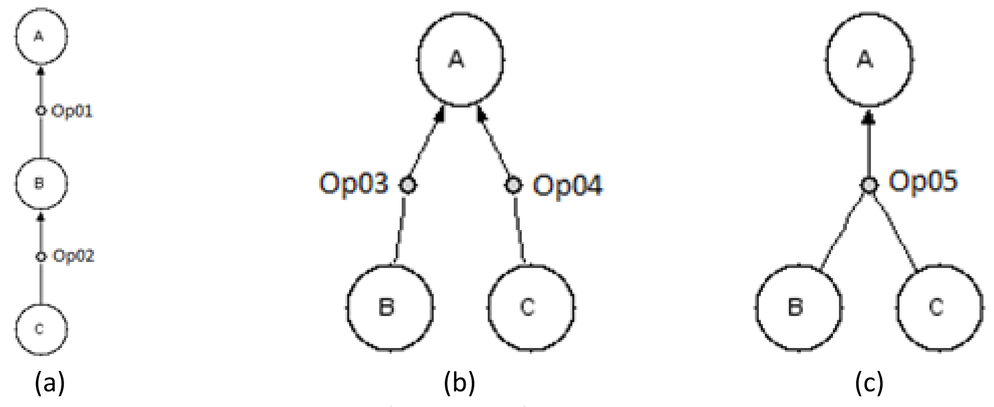


Figure 10 Routing

2.2.2 Event Log

As shown in Figure 5, in order to be able to use process mining techniques, an event log containing relevant process data has to be available. An event log can be extracted from a database, transaction log, audit trails or any other data source that contains information stored by a supporting information system. There are two main formats that are used for event logs: MXML and XES. Both of these formats are supported by the current version of ProM, the framework that will be used for this project.

2.2.2.1 MXML

The Mining eXtensible Markup Language (MXML) format was used as a standard to store event logs and was used by the ProM framework until it got replaced by the XES standard in 2011 as the standard for storing event logs.

The MXML format was first described in [21]. In [22] a distinction is made between actual events and logged events, with the latter being referred to as *audit trail entries*. A number of requirements are stated which should be met by every audit trail entry (ATE):

- Each ATE should be an atomic event that takes place at a certain moment in time
- Each ATE should refer to exactly one uniquely identifiable activity
- Each ATE should contain a description of the event
- Each ATE should refer to a specific process instance (case)
- Each process instance should refer to exactly one process

If all these requirements have been met a correct event log can be generated according to the Meta model[3] given in [22], which describes the structure of the MXML format which is also shown in Figure 11.

[3] The XML Schema definition for MXML is available at: www.processmining.org/WorkflowLog.xsd

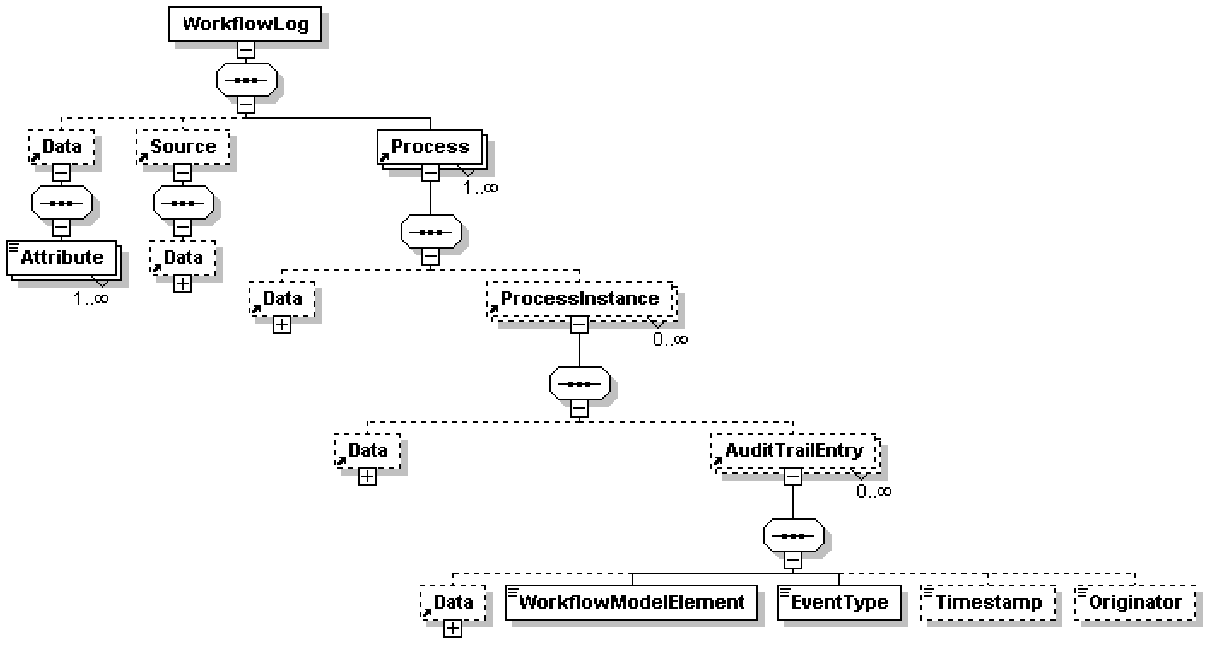


Figure 11 MXML Meta model [22]

Although XES is now the preferred format to use with the ProM framework, the current version of the framework allows for both formats to be used. Since the MXML format has been replaced by the XES format we will not go into further detail here.

2.2.2.2 XES

In 2011 the eXtensible Event Stream (XES) format was selected by the IEEE Task Force on Process Mining as the new standard for storing event logs and has also become the new standard in version 6 (and up) of the ProM framework. The main advantage of the XES format is that it has been made less restrictive than the MXML format. OpenXES is a reference implementation implemented in Java which is used by the ProM framework.

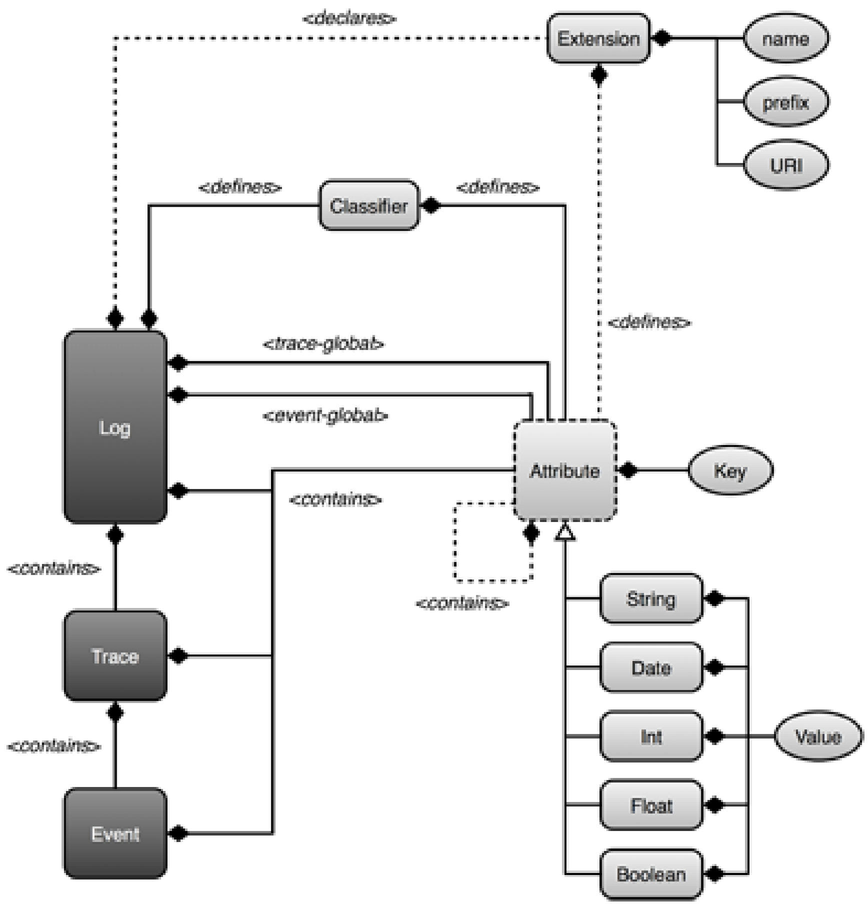


Figure 12 XES Meta model [23]

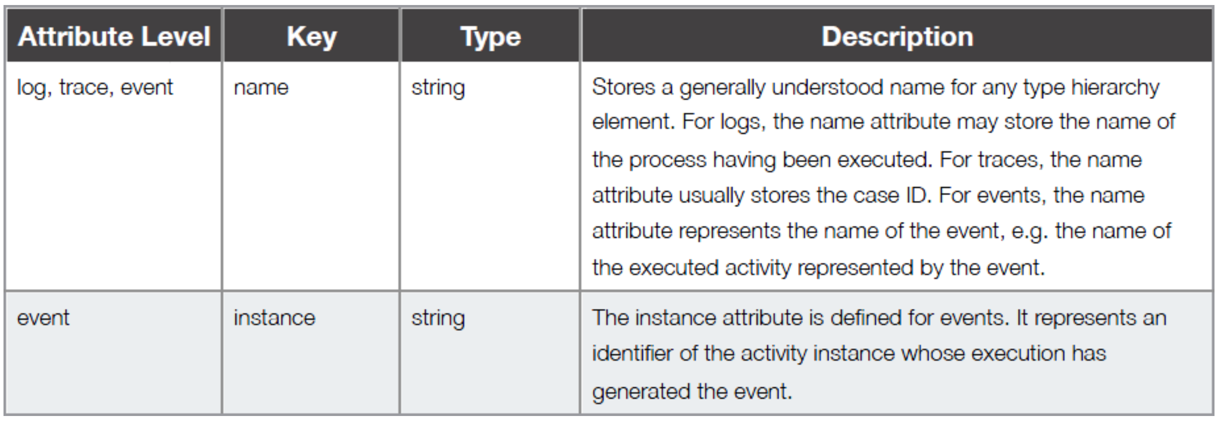
Figure 12 shows the Meta model for the XES format [23]. The XES format allows for one *log* containing a number of traces regarding one specific process. A *trace* consists of all event data concerning one case or process instance. An *event* is an atomic activity which takes place at a certain moment in time during the execution of a process, similar to an audit trail entry. Each of these objects (log, trace and event) can have several *attributes* which describe information about that object. As shown, there are 5 types of attributes defined for the XES format.

Extensions provide a way to add additional information to an event log. Since some information is needed/used for almost every event log, a number of extensions have also been standardized. These standardized extensions are the following:

- Concept
- Lifecycle
- Organizational
- Time
- Semantic

For this project only the Concept and Time extensions are used.

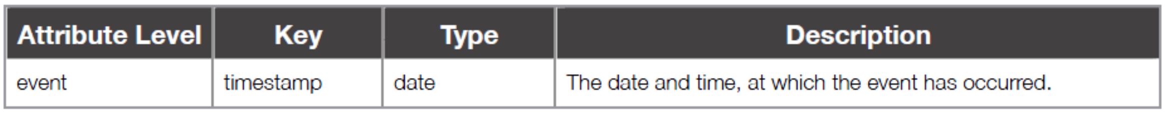
The Concept extension defines an attribute which stores the name of an object. This extension can be used at different levels, which is shown in Figure 13, taken from XES Standard Definition[4] [23].



Attribute Level	Key	Type	Description
log, trace, event	name	string	Stores a generally understood name for any type hierarchy element. For logs, the name attribute may store the name of the process having been executed. For traces, the name attribute usually stores the case ID. For events, the name attribute represents the name of the event, e.g. the name of the executed activity represented by the event.
event	instance	string	The instance attribute is defined for events. It represents an identifier of the activity instance whose execution has generated the event.

Figure 13 Concept extension [23]

The Time extension defines an attribute to store the timestamp at which an event took place. These timestamps are used to derive the order in which the events took place.



Attribute Level	Key	Type	Description
event	timestamp	date	The date and time, at which the event has occurred.

Figure 14 Time extension [23]

An example of a single trace, showing both the Concept and Time extension, is shown below:

```
<trace>
   <string key="concept:name" value="GE42D753427C"/>
   <string key="type" value="Permit Request"/>
   <event>
      <timestamp key="time:timestamp" value="2011-04-26 11:09:57.967"/>
      <string key="requestid" value="D72642830086"/>
      <string key="concept:name" value="Received Request"/>
   </event>
</trace>
```

The trace of this particular case shows that this case with identifier GE42D753427C was a request for a permit. The request, with identifier D72642830086, was received on April 26th, 2011 at almost 10 minutes past 11.

2.2.3 Causal nets

A Causal net (or C-net) is a model in which the nodes represent activities and the connection between these nodes represents causal dependencies. If an activity *A* has to be executed before activity B, then A is the *cause* of B and hence there is a causal dependency between A and B. Since this is also how event logs work, this is a representation that fits well when we want to apply process

[4] Available at: code.deckfour.org/xes/

mining on these logs. Causal nets are a good way to represent dependency relations between objects.

The definition of a Causal net as given in [12] is as follows:

Definition 2 (Causal net (C-net)): *A Causal net (C-net) is a tuple C = (A, a_i, a_o, D, I, O) where:*

- $A \subseteq \mathcal{A}$ is a finite set of *activities*
- $a_i \in A$ is the *start activity*
- $a_o \in A$ is the *end activity*
- $D \subseteq AxA$ is the *dependency relation*
- $AS = \{X \subseteq \mathcal{P}(A) \mid X = \{\emptyset\} \vee \emptyset \notin X\}$
- $I \in A \rightarrow AS$ defines the set of possible *input bindings* per activity
- $O \in A \rightarrow AS$ defines the set of possible *output bindings* per activity

Such that

- $D = \{(a_1, a_2) \in A \text{ x } A \mid a_1 \in \bigcup_{as \in I(a_2)} as\}$
- $D = \{(a_1, a_2) \in A \text{ x } A \mid a_2 \in \bigcup_{as \in O(a_1)} as\}$
- $\{a_i\} = \{a \in A \mid I(a) = \{\emptyset\}\}$
- $\{a_o\} = \{a \in A \mid O(a) = \{\emptyset\}\}$
- All activities in the graph (A,D) are on a path from a_i to a_o

In this definition, a causal net is a set of activities *A* and a (dependency) relation *D*, describing the dependency relations between activities. The C-net has one start and one end element. Given an activity, functions *I* and *O* return the set of activities that are required to have preceded (resp. succeeded) this activity.

Figure 15 shows a visualization of a causal net for a process concerning a request for compensation taken from [12]. On the right hand side the different patterns that can occur in a causal net are displayed.

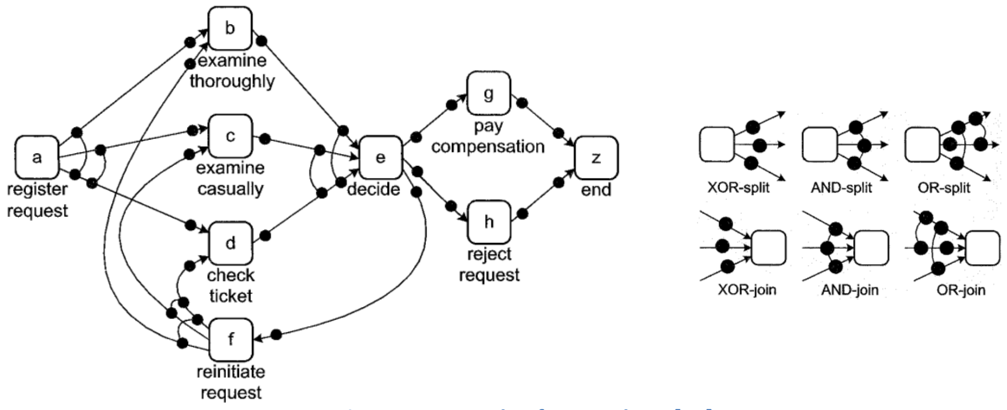


Figure 15 Example of a causal net [12]

2.3 Conclusions

This chapter provided some background information that is useful to better understand the context and content of this project.

The first section introduced a number of artifact-centric approaches used to model processes based on the data that is used during a process rather than focusing on the control-flow. We argued that there is a need to automate the process of generating models in an artifact-centric environment. This way we can speed up this process and generate better models.

Several existing process mining techniques used for activity-based processes are discussed. To illustrate how to apply process mining to document-centric processes, we have chosen PBWD and, in particular the Product Data Model (PDM) used in PBWD, to model these processes.

The second section provided some preliminary information that is necessary in order to be able to better understand the concepts explained in this report. First the Product Data Model (PDM) was described and the formal definition of such PDM was given. Second, the structure of an event log was elaborated and finally an introduction was given in the concept of causal nets. All these concepts will be used in the following chapters.

3 Running example

This section introduces a running example that will be used to illustrate the different steps of the algorithm that are explained in the next chapter.

The running example represents a very simple process of a fictitious car insurance company. This process starts when a claim form is received from a client. This claim is then verified by a claim adjuster who assesses the damage and makes a report of his/her findings. After this report has been made, a review takes place by an underwriter who reviews all available information and determines how much money will be paid to the client and again a report is made. Simultaneously, in cases where the claim is high or the claim is suspicious, a fraud check is done and the history of the client is given a closer look. After this the client is informed that the claim has been accepted and the amount that will be paid.

A	Payment letter
B	Review report
C	Fraud check report
D	Adjuster report
E	Confirmation letter
F	Claim form

Traces	#
FEDCBA	10
FEDBCA	8
FEDBA	25

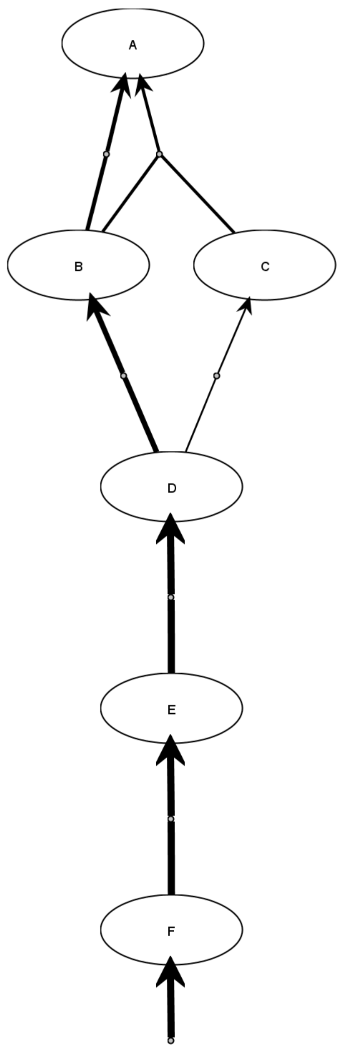


Figure 16 Running example

On the top left of Figure 16 six different documents are shown. The bottom left shows a number of traces containing these documents representing the way the process was executed in the past. It also shows how often each of the traces occurred. On the right a PDM obtained using the traces of the process is shown.

4 Document Log

Recall the framework picture shown in section 1.5:

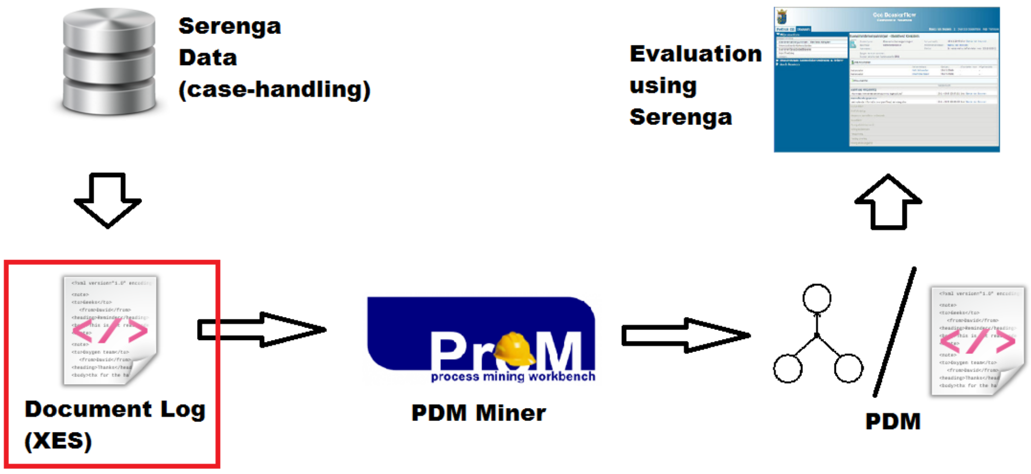


Figure 17 Framework (Document log)

In order to be able to apply process mining techniques a log should be available containing the necessary information. In process mining, the focus is usually on the events or activities in a certain process, hence the use of an event log. In this project however, the focus will be on documents that are used during a process. Therefore, we need different data than one would use for a typical event log. The kind of event log that will be used for this project will therefore be referred to as a *document log*.

In a document log the focus is on the data that is used during a process. Therefore, the main concept of a document log is a *document*. For each document it should be possible to identify a moment in time (timestamp) at which this document was created or added to the process. These timestamps are used to represent the order in which documents where added during a process.

The ProM framework makes use of the XES standard for its event logs. In order for ProM to be able to work with this new document log, this log will also have to conform to the XES standard.

A traditional event log consists of three main concepts that together form the minimal information that is required to create such an event log. These three concepts are:

- *Trace ID* or *Process Instance ID*: unique identifier indicating an instance of a process
- *Event ID* or *Audit Trail Entry (ATE) ID*: indicates a step in the process that was taken during the execution of a process instance
- *Timestamp*: indicates when an activity has taken place

In the case of a document log, there are no activities as such; there are only documents being added to process. Therefore, in a document log, instead of an *event ID,* a *document ID* is used, which identifies a document that is created at a certain point in time (*timestamp*) during the process. As with a traditional event log, both the trace and the documents can have any additional information (besides the IDs).

Document log of running example

Figure 18 shows the document log corresponding to the running example shown in the previous chapter. All three different traces (FEDBA, FEDCBA and FEDBCA) are shown only once here, in the real document log, these traces occur 25, 10 and 8 times respectively.

Each trace has a number of attributes and a number of events. The attributes consist of the trace ID (concept:name), the ID of the process, which in this case is *P01* and the name of the process which is *Claim*.

Each event has an identifier which in this case is the name of the document and a timestamp that indicates when the document was added to the process.

Lifecycles

In order to comply with assumptions 2.1 and 2.2 another assumption has to be made:

Assumption 4.1: *A document can occur at most once in a trace*

For this project it is assumed that a document can occur at most once in each of the traces in a document log. Assumption 4.1 helps to make sure that each document only occurs once in the final PDM after the mining is done and this also minimizes the chance of loops occurring in the final PDM. As we will show later however, loops cannot be entirely removed in all cases, although once a document is used during a process it cannot be used again. Other than this, this assumption does not have a big impact on the quality of the models that are created. This will be demonstrated by some example processes in chapter 6.

As a result of this assumption, if a document is changed during a process after it is added, this has to be filtered out when creating the document log. Either this document has to be split into multiple (versions of the) documents with different identifiers, or one of these changes made to the document has to be chosen to be included in the document log and the rest will be ignored. In the latter case, the best option generally is to take the last time a document was changed as this generally means the document is complete and contains all the information needed during the process.

```
<?xml version="1.0" encoding="UTF-8" ?>
<log xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7"
xmlns="http://www.xes-standard.org/">
        <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
        <extension name="Concept" prefix="concept" uri="http://www.xes-
standard.org/concept.xesext"/>
        <global scope="trace">
                <string key="concept:name" value="__INVALID__"/>
                <string key="processid" value="__INVALID__"/>
                <string key="processname" value="__INVALID__"/>
        </global>
        <global scope="event">
                <timestamp key="time:timestamp" value="0"/>
                <string key="concept:name" value="__INVALID__"/>
        </global>
        <trace>
                <string key="concept:name" value="D0"/>
                <string key="processid" value="P01"/>
                <string key="processname" value="Claim"/>
                <event>
                        <timestamp key="time:timestamp" value="2011-04-26 11:09:57.627"/>
                        <string key="concept:name" value="F"/>
                </event>
                <event>
                        <timestamp key="time:timestamp" value="2011-04-27 10:04:57.326"/>
                        <string key="concept:name" value="E"/>
                </event>
                <event>
                        <timestamp key="time:timestamp" value="2011-04-28 07:02:57.235"/>
                        <string key="concept:name" value="D"/>
                </event>
                <event>
                        <timestamp key="time:timestamp" value="2011-04-29 16:25:57.632"/>
                        <string key="concept:name" value="B"/>
                </event>
                <event>
                        <timestamp key="time:timestamp" value="2011-04-30 08:19:57.823"/>
                        <string key="concept:name" value="A"/>
                </event>
        </trace>
        ...
        <trace>
                <string key="concept:name" value="D25"/>
                <string key="processid" value="P01"/>
                <string key="processname" value="Claim"/>
                <event>
                        <timestamp key="time:timestamp" value="2011-01-13 11:09:57.627"/>
                        <string key="concept:name" value="F"/>
                </event>
                <event>
                        <timestamp key="time:timestamp" value="2011-03-15 10:04:57.326"/>
                        <string key="concept:name" value="E"/>
                </event>
                <event>
                        <timestamp key="time:timestamp" value="2011-05-17 07:02:57.235"/>
                        <string key="concept:name" value="D"/>
                </event>
                <event>
                        <timestamp key="time:timestamp" value="2011-07-19 16:25:57.632"/>
                        <string key="concept:name" value="C"/>
                </event>
                <event>
                        <timestamp key="time:timestamp" value="2011-09-21 08:19:57.823"/>
                        <string key="concept:name" value="B"/>
                </event>
                <event>
                        <timestamp key="time:timestamp" value="2011-11-23 12:46:52.427"/>
                        <string key="concept:name" value="A"/>
                </event>
        </trace>
        ...
        <trace>
                <string key="concept:name" value="D37"/>
                <string key="processid" value="P01"/>
                <string key="processname" value="Claim"/>
                <event>
                        <timestamp key="time:timestamp" value="2010-04-26 11:09:57.967"/>
                        <string key="concept:name" value="F"/>
                </event>
                <event>
                        <timestamp key="time:timestamp" value="2010-10-30 12:46:52.427"/>
                        <string key="concept:name" value="E"/>
                </event>
                <event>
                        <timestamp key="time:timestamp" value="2011-06-01 08:19:57.823"/>
                        <string key="concept:name" value="D"/>
                </event>
                <event>
                        <timestamp key="time:timestamp" value="2011-09-14 16:25:57.632"/>
                        <string key="concept:name" value="B"/>
                </event>
                <event>
                        <timestamp key="time:timestamp" value="2011-06-12 10:04:57.326"/>
                        <string key="concept:name" value="C"/>
                </event>
                <event>
                        <timestamp key="time:timestamp" value="2012-02-29 11:09:57.627"/>
                        <string key="concept:name" value="A"/>
                </event>
        </trace>
</log>
```

Figure 18 Document log of running example

5 Process Mining in a Document-centric environment

This chapter explains the main contribution of this project: how to get from a document log to a Product Data Model. In order to accomplish this, a number of steps need to be. These steps are explained in this chapter.

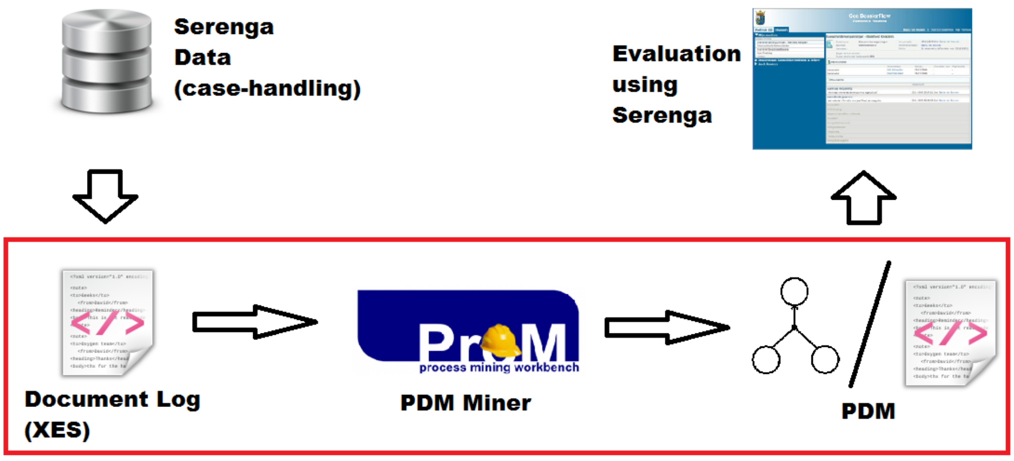


Figure 19 Framework (PDM Miner)

The mining of a PDM from a document log as presented in this section is based on the Flexible Heuristics Miner (FHM) algorithm [18]. The FHM algorithm is normally used to create a process model from an event log and will be modified where necessary to the situation at hand. In this case, we will adapt the FHM algorithm to mine a document log and generate a PDM based on such log.

The formal definition of a PDM, based on [20] and [7] (also shown in section 2.2.1), that will be used for this project is given below.

Definition 3 (Product Data Model): *A Product Data Model (PDM) is a tuple (D, O, root).*

- *D: Set of data elements*
- *O: Set of operations;* $O \subseteq D \times \mathcal{P}(D)$
- $root \in D$

For this project an operation has one additional property which indicates the weight of this operation.

- $weight: O \rightarrow W$, *assigns a weight to an operation*

This property indicates the number of times an operation occurred in a document log relative to the highest occurrence of an operation. Hence, $0 \leq weight \leq 1$. This property will later be used to distinguish common from less common behavior in a model.

As in [12] and [18] Causal nets (or C-nets) will be used to store/represent the data retrieved from a document log. These causal nets are discussed in section 5.1. The rest of the chapter explains how to get from a document log to a PDM. The steps that need to be taken to achieve this are the following:

- Determine direct succession relation (Section 5.2):
  This step determines the order in which documents occur in the log and which documents precede/succeed one another during the process.
- Determine dependency measure (Section 5.3):
  The dependency measure indicates how dependent documents are of one another. This way it is determined which documents are needed in order to create other documents. The dependency measure is also used to filter out uncommon behavior/noise.
- Determining routing information (Section 5.4)
  During the previous steps no routing information is used. This step adds this information combining dependency relations with information available in the log.
- Create PDM (Section 5.5)
  This last step translates the information that resulted from the previous steps into a Product Data Model.

Section 5.6 describes some ways to refine the models that are created during the mining process.

The definitions used in sections 5.2 and 5.3 are based on the FHM algorithm and have been slightly adapted where necessary to work with documents and to be able to create a PDM after the mining is done, resulting in the PDM Miner plug-in.[5]

5.1 Causal nets

The definition of a Causal net given here is based on the definition given in [12] (which can also be found in section 2.2.3):

Definition 4 (Causal net (C-net)): *A Causal net (C-net) is a tuple C = (D, d_i, d_o, DR, I):*

- $D \subseteq \mathcal{D}$ is a finite set of documents
- $D_i \subseteq D$ is a subset of D that contains the documents with which the process starts
- $D_o \in D$ is the final document of the process
- $DR \subseteq D$ x D is the dependency relation
- $I \in D \rightarrow \mathcal{P}(\mathcal{P}(D) \rightarrow \mathbb{N})$ defines the set of possible input bindings per operation

The C-net definition that will be used for this project is slightly different from the one shown in [12]. First, only the input relations (or causes) will be considered. The output relation is not used because one document's output is another document's input and the fact that an operation in a PDM can have only one output element (and multiple input elements). Furthermore, a PDM can have multiple starting points. Therefore, a set of documents as the possible start of the process is used.

Lastly, instead of a set of sets, a bag of sets is used for the input bindings *I* to be able to indicate how often each input binding occurred in the log. In a set no duplicates are allowed, a bag (or multiset) on the other hand does allow duplicates.

When we look at the causal net example given in Figure 15, there is an input binding for element E that consists of both element B and element D (AND-join). Let's say this input binding occurred 25 times in the log. When we would represent this by a set of sets this would just result in the set {{A,B}} which does not show how many times this binding occurred. When representing the same binding

[5] The PDM Miner plug-in will be made available throught the ProductData package of the ProM framework

using a bag of sets we get [{A,B}, {A,B}, …, {A,B}], or [25:{A,B}] for short, as the input binding for element E. This way, we also automatically store information about how often each binding. This will be used later on to distinguish common from uncommon behavior and will be especially useful for the visualization.

5.2 Basic relations

The first step in the algorithm is to check the log to find out which documents appear after one another in the traces. This is done by going through each of the traces and registering which document succeeds another document. In order to do this, the following relation, taken from de FHM algorithm [18], is used:

Definition 5 (Direct succession): *Let D be a set of documents, $\boldsymbol{T}$ the set of all traces over D, $\boldsymbol{\delta} \in \boldsymbol{T}$ is a trace, $\boldsymbol{W}: \boldsymbol{T} \rightarrow \mathbb{N}$ is a document log, and $\boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{D}$:*

- *$a > b$ iff there is a trace $\delta = d_1 d_2 d_3 \dots d_n$ and $i \in \{1, \dots, n-1\}$ such that $\delta \in W$ and $d_i = a$ and d_{i+1} = b (direct successor)*

T is the set of all possible traces containing zero or more documents in D. $W : T \rightarrow \mathbb{N}$ is a bag of traces, hence it can contain a trace multiple times. For this project we do not use the long-distance relations that are defined in [18].

Two documents are in this direct succession relation if they occur after one another in at least one trace in the document log. For each of these relations the number of times it occurs in the log ($|a > b|$) is recorded. As stated in section 3 each document can occur at most once in a trace, which implies the following: if both $|a > b|$ > 0 and $|b > a|$ > 0 (i.e. document *a* can follow document *b* and document *b* can follow document *a*), this means that either this is noise, or *a* and *b* can be executed in any order. The former implies that one of the two relations will have to be ignored (this is done later using certain threshold values), the latter will result in an AND relation. Noise can be filtered out by only accepting direct succession relations that are above a certain threshold, the *positive observations threshold*.

An artificial end element d_{end} is introduced. For each of the documents x that occur at the end of a trace in the log, the counter of the direct successor relation $x > d_{end}$ is increased. This is used to indicate how often a particular document was the last document to occur in a trace. This will later be used to determine the end element of the process. We also keep track of how many times each document occurred at the start of a trace. This information is later used to model the 'start' documents in a model.

Running example (Direct succession measure)

Recall the traces of the running example:

Traces	#
FEDCBA	10
FEDBCA	8
FEDBA	25

Table 2 Traces of running example

If we look at these traces we see for example that E directly succeeds F in all of these traces. The total number of traces is 43, so E directly succeeds F 43 times. Hence, $|\mathrm{E} > F| = 43$. Similar, C directly succeeds B only in the trace FEDBCA, which occurs 8 times in the log. Hence, $|\mathrm{B} > C| = 8$.

Table 3 shows the direct succession matrix for the running example introduced in section 3.

	Start	A	B	C	D	E	F	End
Start	0	0	0	0	0	0	43	0
A	0	0	0	0	0	0	0	43
B	0	35	0	8	0	0	0	0
C	0	8	10	0	0	0	0	0
D	0	0	33	10	0	0	0	0
E	0	0	0	0	43	0	0	0
F	0	0	0	0	0	43	0	0
End	0	0	0	0	0	0	0	0

Table 3 Direct succession matrix

5.3 Dependency Graph (DG)

The next step is to transform the causal net into a *dependency graph (DG).* This dependency graph is constructed using the dependency relations and hence uses only the *(D, d_i, d_o, DR)* part of the C-net. The dependency graph represents the main structure of the C-net and provides a (visual) representation of the dependency relations in which only certain (relevant) relations are shown. Which relations are part of the dependency graph is determined by the so-called *dependency measure* and the values of certain thresholds. The dependency measure of [18] is taken which is defined as follows:

Definition 6 (Dependency measure): *Let L be a document log over D, $\boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{D}$, $|\boldsymbol{a} > \boldsymbol{b}|$ the number of times $\boldsymbol{a} > \boldsymbol{b}$ occurs in L. The dependency measure $\boldsymbol{a} \Rightarrow \boldsymbol{b}$ is defined as:*

$$a \Rightarrow b = \left(\frac{|a>b|-|b>a|}{|a>b|+|b>a|+\sigma_d}\right) \text{ if } (a \neq b)$$

The value of σ_d $(\sigma_d > 0)$ can be used to influence the results of the dependency measure. The higher the value of σ_d, the more occurrences of the documents it takes to get a dependency measure value that will indicate a higher certainty. The default value for σ_d is 1.

The dependency measure $a \Rightarrow b$ will always yield a value *between* -1 and 1, with a value close to 1 indicating that a relation between two documents is very likely. Again note that if both $(a \Rightarrow b)$ <> 0 and $(b \Rightarrow a)$ <> 0, this implies either noise, or an AND relation since *a* and *b* can occur in any order. If one of the values is considerably lower than the other, this might indicate noise and the relation with the lowest value will have to be ignored. If the difference between both values is small, this will most likely indicate an AND relation between the two documents.

Running example (Dependency measure)

As an example let us take the same relations as before. First we take a look at dependency between E and F. As we showed in the previous section, the direct succession relation between E and F resulted in $|\mathrm{E} > F| = 43$. As can be seen from the traces and from the direct succession matrix in Table 3, F is never a direct successor of E and therefore $|\mathrm{E} > F| = 0$. Filling this in in the formula given above results in:

$$\mathrm{E} \Rightarrow \mathrm{F} = \left(\frac{|\mathrm{E} > \mathrm{F}| - |\mathrm{F} > \mathrm{E}|}{|\mathrm{E} > F| + |\mathrm{F} > E| + 1}\right) = \left(\frac{0 - 43}{43 + 0 + 1}\right) = \left(\frac{-43}{44}\right) = -0.977273$$

Similar,

$$\mathrm{F} \Rightarrow \mathrm{E} = \left(\frac{|\mathrm{F} > \mathrm{E}| - |\mathrm{E} > \mathrm{F}|}{|\mathrm{F} > E| + |\mathrm{E} > F| + 1}\right) = \left(\frac{43 - 0}{0 + 43 + 1}\right) = \left(\frac{43}{44}\right) = 0.977273$$

This indicates that E depends on F and not the other way around.

When looking at the dependency measures of B and C, we get that:

$$\mathrm{B} \Rightarrow \mathrm{C} = \left(\frac{|\mathrm{B} > \mathrm{C}| - |\mathrm{C} > B|}{|\mathrm{B} > C| + |\mathrm{C} > B| + 1}\right) = \left(\frac{8 - 10}{8 + 10 + 1}\right) = \left(\frac{-2}{19}\right) = -0.105263$$

And again the other way around,

$$\mathrm{C} \Rightarrow \mathrm{B} = \left(\frac{|\mathrm{C} > \mathrm{B}| - |\mathrm{B} > C|}{|\mathrm{C} > B| + |\mathrm{B} > C| + 1}\right) = \left(\frac{10 - 8}{10 + 8 + 1}\right) = \left(\frac{2}{19}\right) = 0.105263$$

In this case, the low dependency measure (close to 0) indicates that B and C can occur in any order in the trace and that both cases occur a similar amount of times in the traces. That is, there is a dependency between the two (given the direct succession values), but both documents can appear in any order. Hence, B and C are executed in parallel instead of after one another.

Table 4 shows the dependency measures for the running example in matrix form.

	Start	A	B	C	D	E	F	End
Start	0	0	0	0	0	0	0.977273	0
A	0	0	-0.972222	-0.888889	0	0	0	0.977273
B	0	0.972222	0	-0.105263	-0.970588	0	0	0
C	0	0.888889	0.105263	0	-0.909091	0	0	0
D	0	0	0.970588	0.909091	0	-0.977273	0	0
E	0	0	0	0	0.977273	0	-0.977273	0
F	0	0	0	0	0	0.977273	0	0
End	0	-0.977273	0	0	0	0	0	0

Table 4 Dependency measures matrix of running example

To filter the dependencies that will be used in the DG several thresholds and settings can be used:

- **Positive observations threshold:** Used to distinguish noise from common (parallel) behavior.
- **Dependency threshold:** This is the main threshold indicating which values for $a \Rightarrow b$ will be allowed. For example, a dependency threshold of 0.9 indicates that all $a \Rightarrow b$ relations with a value over (or equal to) 0.9 will be used in the creation of the DG.
- **All-tasks-connected (ATC) heuristic:** basically selects the document(s) with the highest $a \Rightarrow b$ value for each of the documents. This way each document will be included in the final model and each document has at least one other document connected to it.
- **Relative-to-best threshold:** This threshold can be used in combination with the all-tasks-connected heuristic. When using this threshold, in addition to the highest $a \Rightarrow b$ value, also documents that have a dependency threshold that is close to this highest value will be included.

The next table shows the steps of the *Dependency Graph* (DG)-algorithm (similar to the algorithm in [18], but without loops). The algorithm creates an artificial start and end element for each trace. Especially the artificial end is useful later on when determining a root element for the PDM. This table assumes that the all-tasks-connected heuristics is used together with a relative-to-best threshold of σ_r.

	$\boldsymbol{W}$ ***data element log over*** $\boldsymbol{D, W^+ = \{start\ \delta\ end\ \vert \delta \in W\}}$
1	$D = \{d \mid \exists_{\sigma \in W^+}[d \in \sigma]\}$
2	$C_{out} = \{(a,b) \in D \times D \mid a \neq End \wedge a \neq b \wedge \forall_{y \in D}[a \Rightarrow_W b \geq a \Rightarrow_W y]\}$ (strongest follower)
3	$C_{in} = \{(a,b) \in D \times D \mid b \neq Start \wedge a \neq b \wedge \forall_{x \in D}[a \Rightarrow_W b \geq x \Rightarrow_W b]\}$ (strongest cause)
4	$C''_{out} = \{(a,b) \in D \times D \mid (a \Rightarrow_W b) \geq \sigma_a \vee \exists_{(a,c) \in C_{out}}[((a \Rightarrow_W c) - (a \Rightarrow_W b)) < \sigma_r]\}$
5	$C''_{in} = (b,a) \in D \times D \mid (b \Rightarrow_W a) \geq \sigma_a \vee \exists_{(b,c) \in C_{in}}[((b \Rightarrow_W c) - (b \Rightarrow_W a)) < \sigma_r]\}$
6	$DG = C''_{out} \cup C''_{in}$

Table 5 Dependency graph algorithm with ATC heuristic

In step 1 all distinct documents are determined. In step 2 and 3 for each document $\boldsymbol{x}$, the strongest follower (document $\boldsymbol{y}$ with highest $\boldsymbol{x} \Rightarrow \boldsymbol{y}$ value) and strongest cause (document $\boldsymbol{z}$ with the highest $\boldsymbol{z} \Rightarrow \boldsymbol{x}$ value) are selected (that is, if the all-tasks-connected heuristic is used). Note that there can be more than one strongest follower/cause. These strongest followers/causes are used in steps 4 and 5. Step 4 and 5 use the thresholds mentioned above ($\boldsymbol{\sigma_a}$ = dependency threshold and $\boldsymbol{\sigma_r}$ = relative-to-best threshold) to select (additional) relations that will be used to create the DG. Step 6 combines the results of the previous steps to create the DG.

When not using the all-tasks-connected heuristic (and hence the relative-to-best threshold), there are only 4 steps that need to be taken:

1	$D = \{d \mid \exists_{\sigma \in W^+}[d \in \sigma]\}$
2	$C''_{out} = \{(a,b) \in D \times D \mid (a \Rightarrow_W b) \geq \sigma_a\}$
3	$C''_{in} = \{(b,a) \in D \times D \mid (b \Rightarrow_W a) \geq \sigma_a\}$
4	$DG = C''_{out} \cup C''_{in}$

Table 6 Dependency graph algorithm without ATC heuristic

Running example (Dependency graph)

For this running example we will use the all-tasks connected heuristic and a dependency threshold of 0.60). Although the all-tasks-connected heuristic is not really necessary it is still used to be able to go through all the steps given above.

The first step is to get all documents:

1. D = {Start,A,B,C,D,E,F,End}

The next step is to select the strongest followers (C_{out}) and strongest causes (C_{in}) for each of the documents. These can easily be determined when looking at the dependency measures matrix. For the strongest causes we look at the highest value for each column, the strongest follower is found by looking at the highest value for each row. For example, the strongest cause of D is document E ($\mathrm{E} \Rightarrow \mathrm{D} = 0.977273$) and the strongest follower is document B ($\mathrm{D} \Rightarrow \mathrm{B} = 0.970588$).

2. C_{out} = {(Start,F),(F,E),(E,D),(D,B),(C,A),(B,A),(A,End)}

3. C_{in} = {(Start,F),(F,E),(E,D),(D,C),(D,B),(B,A),(A,End)}

In step 4 and 5 these sets are combined with additional causes and followers that exceed the selected dependency threshold. In this case we will select a dependency threshold of 0.6.

4. C''_{out} = {(Start,F),(F,E),(E,D),(D,C),(D,B),(C,A),(B,A),(A,End)}
5. C''_{in} = {(Start,F),(F,E),(E,D),(D,C),(D,B),(C,A),(B,A),(A,End)}

Combining these last two sets results in the final dependency graph:

6. DG = {(Start,F),(F,E),(E,D),(D,C),(D,B),(C,A),(B,A),(A,End)}

Table 7 shows the DG in matrix form that contains that part of the dependency measures from Table 4 that exceed a dependency threshold of 0.6.

	Start	A	B	C	D	E	F	End
Start	0	0	0	0	0	0	0.977273	0
A	0	0	0	0	0	0	0	0.977273
B	0	0.972222	0	0	0	0	0	0
C	0	0.888889	0	0	0	0	0	0
D	0	0	0.970588	0.909091	0	0	0	0
E	0	0	0	0	0.977273	0	0	0
F	0	0	0	0	0	0.977273	0	0
End	0	0	0	0	0	0	0	0

Table 7 Accepted dependency measures with dependency threshold set to 0.60

Figure 20 shows the graphical representation of the dependency graph (ignoring the artificial start and end). The numbers at the arcs indicate how often a certain dependency occurred and what the dependency measure is between two documents.

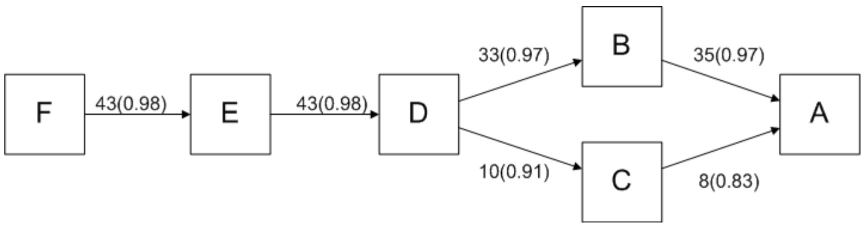


Figure 20 Dependency graph of running example

5.4 Routing information

Executing the steps shown above results in a dependency graph, however, this dependency graph has no routing information yet. This section will show how to include routing information.

This routing information is determined by using the generated DG and the document log to replay the log. Doing so, function I of the C-net can be derived and the input bindings can be set. As mentioned before, to be able to use the number of occurrences of each input binding, bags of sets will be used instead of sets of sets. This way it is possible to count the number of times each input binding occurs in the document log. This will be important later on when the resulting PDM will be visualized.

To determine the input bindings, two steps need to be taken for each trace in the document log. First the dependency relations for a document are retrieved from the (original) C-net, stating which documents are causes of the current document; these documents will be referred to as the *input set* of a document.

Running example (input sets and input bags)

When we look at the running example, we have the input sets as shown in Table 8. For example, document F was the only possible cause of document E. Therefore, the input set of F consists of only document E. Document A on the other hand had two possible causes, document B and document C, hence the input set of document A consists of both B and C.

Input set	Document
{B,C}	A
{D}	B
{D}	C
{E}	D
{F}	E
{ }	F

Table 8 Example input sets

However, from these input sets it is not clear whether A is always the result of B and C for all of the cases or that A can also be the result of B in one case and C in another case, or a combination of both. This information is however stored in the log. That is why we need to take a look at these input sets in combination with the traces in the document log to derive this kind of routing information.

To do this, we derive so called input bags for each of the documents. What we basically do with these input bags is count the number of times a certain subset of the input set was the cause of a document. Using this information we can determine the routing. We will explain this using the running example.

Recall the traces of the running example again:

Traces	#
FEDCBA	10
FEDBCA	8
FEDBA	25

Table 9 Traces of running example

Now we know from the input sets that we only accept B and C as causes of A (any other causes of A may have been filtered out already). Now we need to know in how many traces A was the result of B, in how many traces A was the result of C and in how many traces A was the result of both B and C.

We start out with an empty input bag for document A. In the trace FEDCBA that occurred 10 times in the document log, both B and C occurred before A. From this we conclude that in these cases A was the result of both B and C. Hence, we update the input bag of A to include 10 times the set {B,C}, which results in an input bag of [10:{B,C}].

Next we have the trace FEDBCA. Similar to the previous trace, we have that A was the result of both B and C, only this time this trace only occurred 8 times. Hence, we update the input bag of A to include 8 times the set {B,C}, which now results in an input bag for A of [18:{B,C}].

The last trace, FEDBA, shows that A was the result of only B (since the trace does not include document C). This trace occurred a total of 25 times and hence we update the input bag again to include the set {B} 25 times.

As a result the input bag of A becomes [18:{B,C},25:{B}]. This shows us that in 25 cases only document B was needed as input to create document A and in 18 cases both document B and document C where needed to be able to create document A.

This is repeated for each document, resulting in the input bags shown on the right side of the figure below.

Traces	#
FEDCBA	10
FEDBCA	8
FEDBA	25

Input set	Document
{B,C}	A
{D}	B
{D}	C
{E}	D
{F}	E
{}	F

$\Rightarrow$

Input bag	Document
[18:{B,C},25:{B}]	A
[33:{D}]	B
[10:{D}]	C
[43:{E}]	D
[43:{F}]	E
[]	F

Figure 21 Input bags for running example

Before determining these input bags, we did not know what exactly was needed in order to be able to create document A. Now we now that in most cases, document B is sufficient to be able to create document A, however, in a lot of cases, both document B and document C are needed. As we know, these are the 'high-risk' cases in which an extra step is made to check for fraudulent behavior.

Figure 22 shows a visual representation of the resulting Causal net in which the routing is also processed. The construction between documents B, C and A shows that A can be the result of either document B, which occurred 25 times or it can be the result of both document B and document C, which occurred 18 times in the document log.

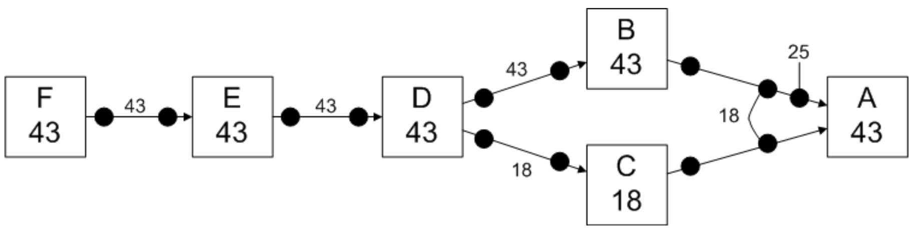


Figure 22 Causal net of running example using dependency threshold of 0.6

5.5 Mapping C-net to PDM

Now that the different relations and bindings have been mined the C-net has to be mapped to a PDM. Since not all elements of the PDM as defined in Section 2.2.1 are needed, an adapted version of the PDM (with only (D, O, root)) is used as was mentioned at the beginning of this chapter. The mapping of a C-net (D_C, d_i, d_o, DR, I) to this adapted PDM (D, O, root) is done as follows:

1. Add documents in C-net to PDM
   - *D* = D_C
2. Create operations (O) according to C-net
   - *Let* $I_C(d)$ *the input bag from the C-net for a given document* d*, then, for each* $is \in I_C(d)$*:*
   - $O = O \cup \{(d, is) \in D_C \times \mathcal{P}(D_C)\}$
3. The root is set to the element that is determined as described in section 5.6.1
   - *root* = d_o

Step 1 copies all documents that are used in the C-net to the PDM. In the second step, the input sets are translated to operations in the PDM. For each input set in a bag, an operation is created accordingly, with the documents of the input set as input or the operation and the document the bag belongs to as the output document.

Table 10 shows the input relations of the C-net used to create the model for our running example. In the case of document A for example, there are two sets in the input of document A. When mapping this to the PDM, this results in two operations: ({A},{B,C}) and ({A},{B}).

Input bag	Document
[18:{B,C},25:{B}]	A
[33:{D}]	B
[10:{D}]	C
[43:{E}]	D
[43:{F}]	E
[]	F

Table 10 C-net used for running example

As mentioned in section 2.2.1, each operation also has a weight assigned to it. As one operation is created for each input set, this weight is based on the number of occurrences of such an input set and is defined as follows:

- $weight: O \rightarrow W$, *assigns a weight to an operation*

The weight of an operation *O* is equal to the weight of the set *S* and therefore:

$$weight(O) = weight(S) = (\#occurences\ of\ input\ set\ s\ in\ input\ bag)/w_{max}$$

Where w_{max} is the maximum size of an input bag in the model (biggest input bag).

The last step is to set the root element of the PDM to the element that is selected as the root by the mining algorithm. In most cases there is a clear end element or a goal of a process. However, a process might have more than one way to terminate. This can either be an already existing document or an artificially created end of the process as we will see next.

5.6 Model refinement

Applying the algorithm as described above, even when using the different thresholds, may still result in a model that is too complex to understand. Furthermore, some PDM specific points need to be taken into account to come to a correct model.

5.6.1 Root element(s)

With the creation of the C-net, an artificial end element is created. This artificial end element is only useful when there would otherwise be multiple elements with which the process could terminate. When the artificial end element has only 1 document in its input relation, the artificial element can be removed, leaving the document in its input relation as the end element. If there are however multiple documents in the input relation of the artificial end element, this element should remain as the 'root' element, because removing it would also remove some restrictions on the model.

A PDM allows for only one root element. However, some processes can have multiple finishing possibilities. Let's take our running example. In our case we only looked at claims that were approved. In real life however, claims can also be rejected and hence a rejection letter is created and sent to the person that filled out the claim form. For this, another document is added (element X) that indicates that a letter was sent rejecting the claim. This way we get a process in which a trace can end with two possibilities, either element F (a letter is sent in which the claim is accepted and an in which the amount is stated) or element X (a letter is sent stating the claim is rejected). In this situation we leave the artificial end element and connect all the elements in the input bag to this end element (XOR relation), in this case element F and element X.

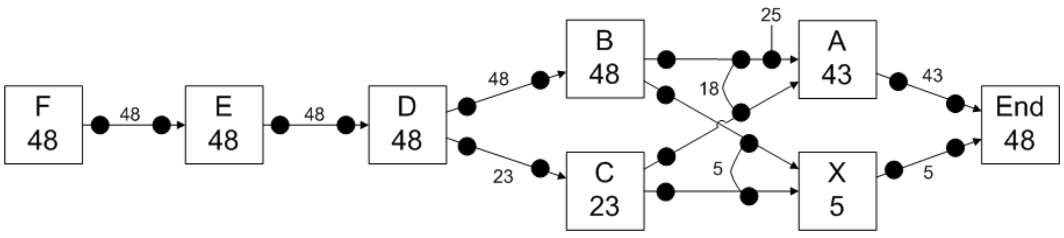


Figure 23 Artificial root element

Figure 23 shows this altered C-net with the artificial root element which indicates that a case is closed.

5.6.2 Redundant input bindings

In some situations redundant operations might occur. For some processes, the actual document is not used in an operation, but only the availability of the document is required. In chapter 6 for example, document groups are used instead of actual documents. These document groups become complete when the last document is added. Hence, an operation is only enabled if all its input document groups are complete.

Figure 24 shows a small example using this document group principle. The document *'Permit'* in Figure 24 has two sets in its input bag: *{Confirmation}* and *{Permit Request, Confirmation}*. But as can be seen from Figure 24a, *'Confirmation'* in turn (only) has *'Permit Request'* in its input bag. Hence, when determining a value for *'Permit'*, a value for *'Permit Request'* is already available, since it was needed to determine a value for *'Confirmation'*. In this case, this makes *'Permit request'* in *{Permit Request, Confirmation}* redundant and *'Permit Request'* can therefore be removed from this set. Consequently, this makes the whole set redundant as the remaining *'Confirmation'* can be merged with the already existing set *{Confirmation},* resulting in the model shown in Figure 24b.

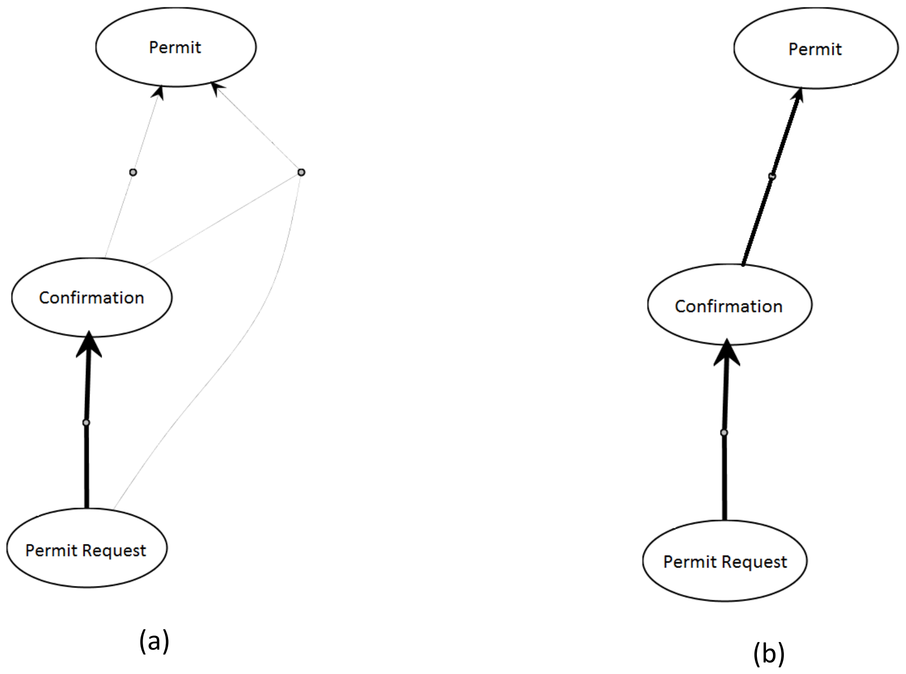


Figure 24 Example of redundant operation
a) Model containing redundant operation, b) Model with redundant operation removed

Although, in itself, this is not wrong, it provides redundant information which results in a redundant operation (extra arcs that might clutter the model), which can be left out of the model.

In the case shown in Figure 24 the documents that are in an AND relation (two or more documents in one input binding of a document), *'Permit Request'* and *'Confirmation'*, are directly linked to each other by another operation (*'Permit Request'* is the direct input of *'Confirmation'*). However, this does not necessarily have to be the case; there can be an arbitrary number of documents/operations between two documents in an AND relation. To remove these kinds of redundant relations, all AND input sets (of all documents) will have to be checked.

Let X be a document and let $S = \{A, B, \dots\}$ be an AND input set of X. For each two documents in this input set we need to check whether or not there is a path in the model from document A to document B or vice versa (using the input relations). If so, we need to find out if there are other ways to reach document B. If there is no possibility to reach document B without using document A, (i.e. no path from a start element to document B exists that does not contain A), document A can be removed from S. If there is such a path, this means that document B can be assigned a value without needing document A and hence the AND relation will have to remain in order to restrict the creation of the document which has the AND relation as its input relation.

Figure 25 shows two situations that can occur. In Figure 25a a situation is shown in which a redundant relation can be removed. We assume that there are no XOR relations on the path from document Y to document B. In this situation all paths to document B contain document A (the path is $AZ \rightarrow Y \rightarrow \cdots \rightarrow B$). Figure 25b shows a situation in which the relation cannot be removed. Here, two choices/paths are possible to create document Y; either using document A ($A \rightarrow Y \rightarrow \cdots \rightarrow B$) or document Z ($Z \rightarrow Y \rightarrow \cdots \rightarrow B$). If the latter path is chosen, document Z is used to create document Y, and hence, document A will not be part of the path to document B. In this situation the AND relation between document A and document B is relevant and cannot be removed.

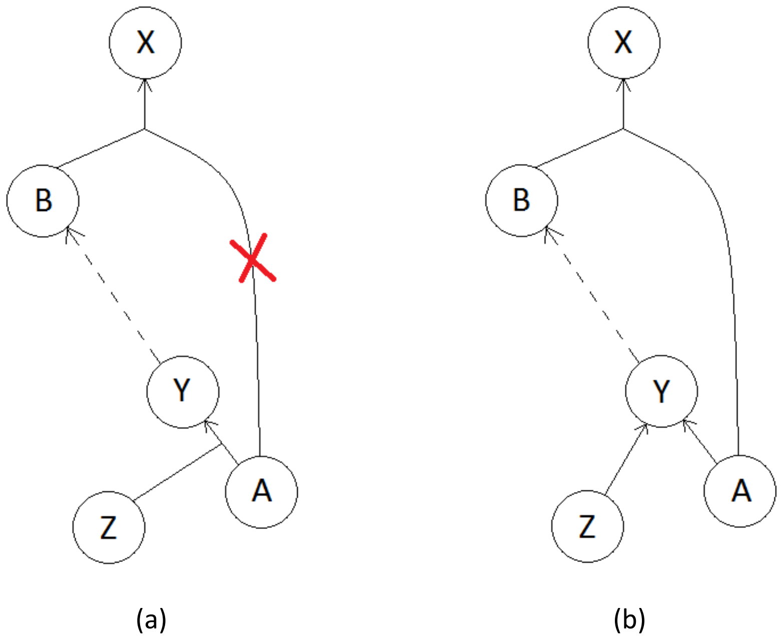


Figure 25 Redundant relation

5.6.3 Low frequent behavior

In the FHM approach there are no thresholds that are applied after the bags have been created. This way, low frequent behavior (or noise) can still appear in the bags and, as a result, in the model. For example an input set of {B,C,D} may result in an input bag like [5:{B,C}, 210:{B,C,D}]. One might not be interested in the low frequent behavior, but only in the general path that is taken in which case the input from {B,C} can be disregarded.

For this, another threshold is introduced to reduce the complexity of the model. Using this threshold one can select which operations are shown in the PDM. This option ensures that only the operations with the highest number of occurrences (higher than the threshold) are shown in the PDM. A higher threshold thus results in only the most common behavior being shown in the model. This is implemented by sorting the sets on the number of occurrences (highest to lowest) and selecting sets as long as the number of occurrences is higher than the threshold.

This threshold can also be used in combination with the all-tasks-connected heuristic of the FHM algorithm, in which case connections are not removed if this may lead to an 'unconnected' element. This is done by leaving (at least) one connection for each document in the input set of a document. For the implementation, again the sets are sorted. While going through the sets, it is registered if a document has already been included in the resulting bag. All sets above the threshold are included again. But if after this not all documents are included yet, the next set (highest occurrence below the threshold) containing such document is included also, until all documents are contained in the new input bag.

Example 5.1: Given the input bag [7:{A,B}, 1:{A,C,D},4:{C},21:{B,D}] and the threshold set to 6. If all-tasks-connected is not enabled then the result of this step will in this case be [21:{B,D},7:{A,B}]. If however the all-tasks-connected option is enabled, the resulting bag will be [21:{B,D},7:{A,B},4:{C}]. Figure 26 shows the different resulting models.

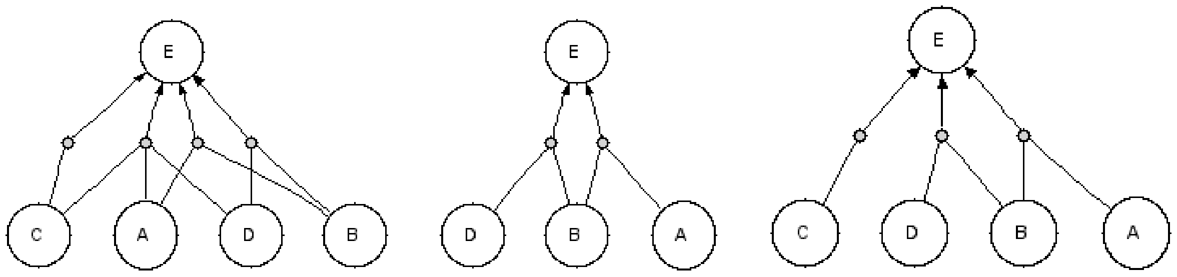


Figure 26 Different levels of detail:
The left model shows the original model. The model in the middle shows the adapted model without the all-tasks-connected option. The model on the right shows the adapted model with the all-tasks-connected option enabled.

6 Evaluation: Applying the algorithm to Serenga data

The algorithm as described in the previous section generates a PDM based on a document log that contains information about documents.

As mentioned in chapter 1 this project was done in collaboration with the R&D department of Océ Technologies. The data that was provided for this project consists of usage data generated by Serenga, a case management system that is used by customers to archive and manage their data-driven processes. In this chapter we use this data to evaluate the PDM Miner with real usage data.

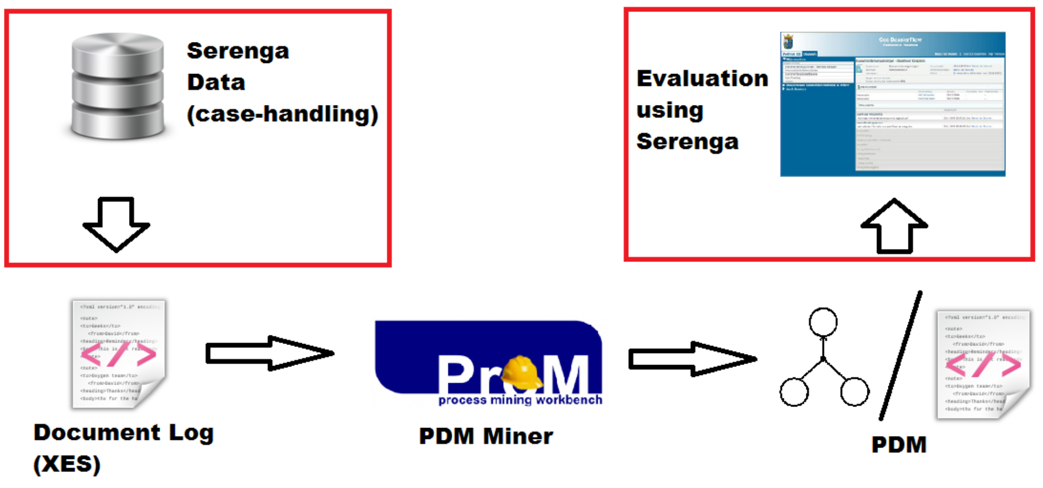


Figure 27 Framework (Serenga data)

The first sections explain how the data from Serenga is used to create a document log and generate process models for the processes that are executed using Serenga. Section 6.5 introduces a measure, the successful execution measure, that is used to indicate how well the PDM Miner performs. Section 6.6 shows the results of applying the PDM Miner to a number of processes which are actual processes of one of Océ's customers, a Dutch municipality, that uses Serenga for its processes.

6.1 Selection of the right data

As stated before, the data that will be used for the document log is retrieved from usage information from a Serenga database/logging. In Serenga, the main entity (or case) is a dossier, which consists of a number of documents and activities.

For this project a dossier is defined as a tuple (D, A):

- *D: Set of documents*
- *A: Set of activities*

However, these documents and activities are not linked to each other.

Once a dossier has been created, a couple of actions can be taken (with respect to documents):

- A document can be added to a dossier
- A document can be viewed
- A document can be changed
- A document can be removed from a dossier

With this structure in mind, there are some limitations as to what information can be used for the mining process and hence which data is useful to be used in the document log. For the data set that is provided, the smallest possible unit that can be used for the mining process is the group of documents (Document group in Serenga). A document group is a collection of documents and hence, is more a container concept which contains documents. For each dossier, a document group can contain zero or more documents. The last addition of a document to a certain document group would indicate that every document for this document group has been received and hence the timestamp of the last addition of a document to a document group will be used to indicate that a document group is complete and this will be the timestamp that will occur in the document log for this document group.

In Serenga, documents also have a filename (besides an ID) which can be used for identification. However, in the case of the municipality for example, the filenames that are used are not a closed or predefined set of names. This means that for each dossier, the filenames of (similar) documents can be different. When looking at a process that handles the approval of permits, a letter containing a request for a permit can be named '20110426183467340 request permit event X.pdf' in one case, while in the other the request is named 'Permit request for association Y.pdf'. This makes it impossible to use the documents themselves as main elements for the mining process.

Taking it one step further, activities and their results (possible next steps to be taken) are based on information inside these documents. The ideal data would thus be the bits of information inside the documents which are used for decision making during a process. However, extracting this information from the documents that are stored cannot be automated since it is not possible to get this information without someone indicating what to look for and even then it would possibly require complex work to retrieve this information from the documents. Also, the rules that are used for decision making are not explicitly stored. Hence, there is no way to use information inside documents.

Of course there may be other customers that do use standardized names for their filenames (for example: 'requestpermit.pdf' for a certain dossier type or even for all dossiers) or some other way to relate similar documents, so that these can be used to generate a document log. Or one might even have standardized forms as documents that can be mined. The algorithm can handle any of these situations; the document log however should then contain different information, i.e. the identifier (or concept:name) attribute should be assigned accordingly. But as indicated before, for now this leaves the use of the document group for the document log.

6.2 Creating the document log

As indicated in the previous section, a dossier consists of a number of document groups, which in turn contain a number of documents that are added to this dossier at a certain moment during the process, which is indicated by a timestamp. This information will be used to construct a document log which in turn will be translated into a PDM (Section 6.3). The information that will be used to create a document log is shown in Table 11.

Attribute	Description
DossierID	The identifier of the dossier
Timestamp	Timestamp of when a document was added
DossierTypeID	The identifier of the dossier type
DossierType	A textual description (name) of the dossier type
DocumentID	The identifier of the document
DocumentGroup	A textual description (name) of the document group/group the document belongs to
Action	The action that was taken on the document

Table 11 Document log

Table 12 shows an example of this information with Serenga data:

Attribute	Value
DossierID	E7B4E50F-BE5F-4CB7-B25C-D43A152D6A37
Timestamp	2009-09-02 13:29:32.577
DossierTypeID	B0056
DossierType	Evenementenvergunningen
DocumentID	4E2C5EDF-7BA2-4BF5-A235-DB36C5ADE957
DocumentGroup	Aanvraag vergunning
Action	dossier gecreëerd op basis van intake document

Table 12 Example data from Serenga

The usage data of Serenga is stored in a MSSQL database. This data first needs to be filtered to get the right information that is required to build the document log.

Each dossier is represented by a trace, with *DossierID* as its main identifier (trace ID). Other information stored at the trace level is the *DossierTypeID* and the *DossierType*. The *DossierType* is later used to select one dossier type for which a model will be generated. The *Action* attribute is used to select only the last addition (of a document) to a certain *DocumentGroup*. Each final document addition of each *DocumentGroup* is translated into an entry in the log with the corresponding *Timestamp*.

The result from filtering the data is then transformed into an XES (.xes) file.

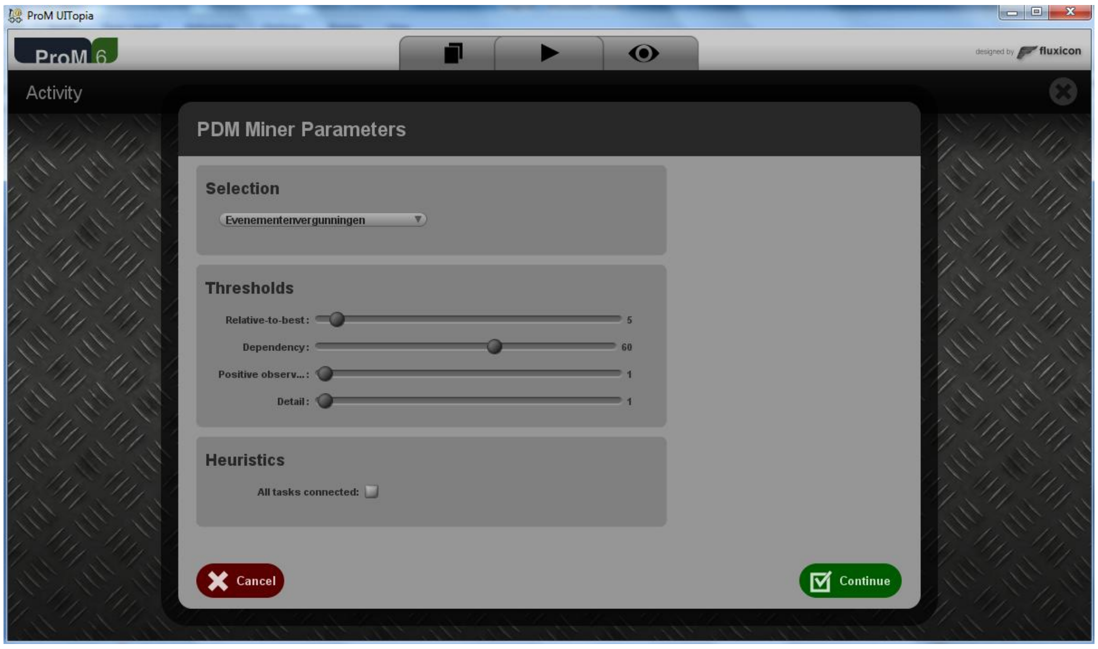
6.3 Creating a PDM

Since in this case only document groups are used we use a stripped down version of the PDM as defined in section 2.2.1, consisting of only *(D, O, root).* This adapted PDM will be used to model the processes executed with Serenga.

Recall that in this adapted PDM, the operations have one additional property that is the weight of the operation. This weight is based on the number of times the operation was executed in the log. The weight of an operation is therefore defined as the weight of the input set it was based on. This property is used to visualize which operations are executed more than others: the higher the weight, the thicker the arrow in the model. In this way the users of Serenga can easily see what paths in the process are used the most and which paths are rarely taken and hence, a distinction can be made between common and uncommon behavior.

6.4 Implementation

The algorithm described in this report is implemented as a plug-in for the ProM framework and is thus implemented in JAVA. Figure 28 shows the settings window for the plug-in in the ProM framework. Figure 29 shows the main visualization window containing a visualization of the PDM.


ProM UITopia

Activity

PDM Miner Parameters

Selection

Evenementenvergunningen

Thresholds

Relative-to-best: 5

Dependency: 60

Positive observ...: 1

Detail: 1

Heuristics

All tasks connected: ☐

Figure 28 PDM Miner settings in ProM

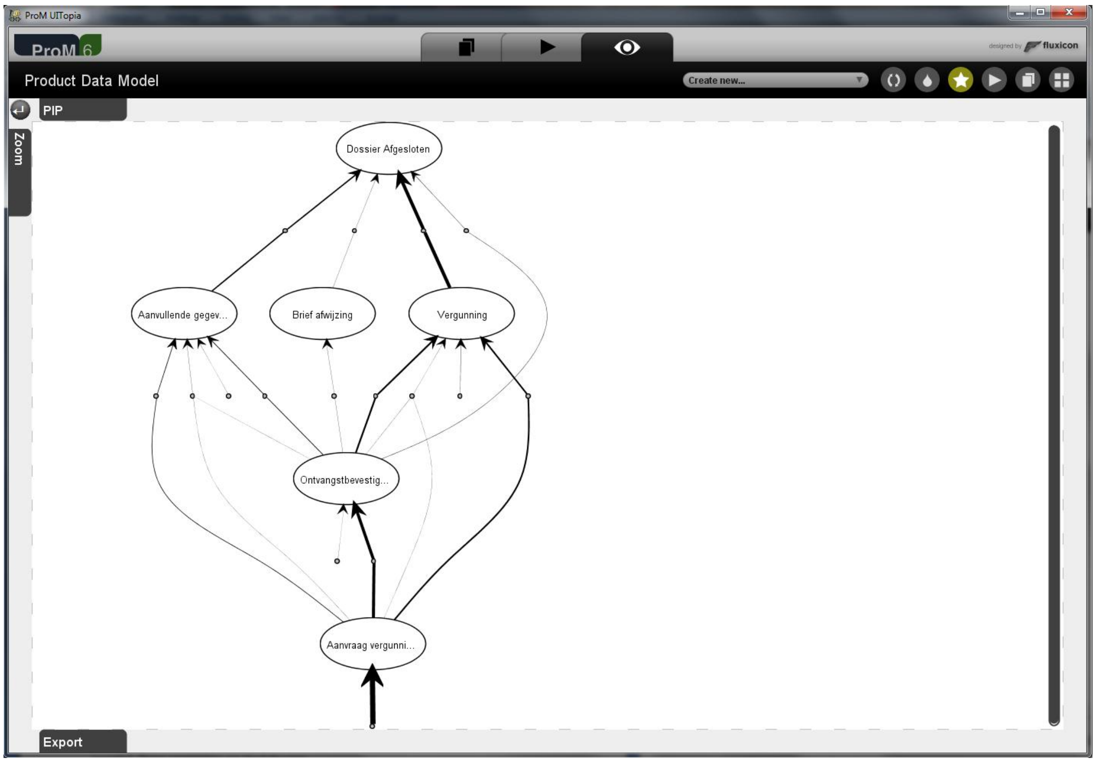


Figure 29 Visualization window in ProM showing a PDM

One of the requirements of Océ was to be able to automatically create a log from database data and afterwards run the algorithm automatically so that both could be scheduled to run without any user interaction. This meant that it should be possible to run the ProM plug-in through the command line interface. In order to do this several steps were to be taken.

6.4.1 Creating the document log

First, retrieving the required data from the database is separated from the actual mining plug-in in a separate (runnable) JAR file. In order to execute this JAR file, a couple of parameters need to be supplied. These parameters are the following:

- domain: the domain of the MS SQL server
- dbname: the name of the database that contains the relevant data
- output folder: the folder in which the XES logs will be stored

Using this information, the data is retrieved from the specified database using single sign-on (using Windows Authentication) and is put in the XES format (described in Section 2.2.2.2) after which an XES file is stored. One XES file is created for each of the dossier types that are present in the database. These files are put in a new directory called 'logs' that is created in the output folder provided by the user and will be used as input for the mining of the PDMs. Furthermore, these files can also be used as input for the ProM framework (using the GUI) so that additional mining in ProM can be done.

6.4.2 Mining the log

The next step is the actual mining. The mining algorithm is put in a separate JAR file so that both the creation of document logs and the mining can be done independently of one another. Again some parameters have to be passed along:

- log folder: Can either be one single XES file (ending with ".xes") or a directory containing one or more XES files
- number of thresholds: The number of thresholds for which a model will have to be created (and hence, the number of models that will be created)
- dependency thresholds: The actual thresholds that are used for the mining of the models (equal to the number of thresholds specified)

If a single XES file is specified than only this single XES file will be mined, if a folder/directory is specified all XES files in this folder will be used and a PDM will be created for each of these logs. After the mining is done, two separate folders are created inside the log folder:

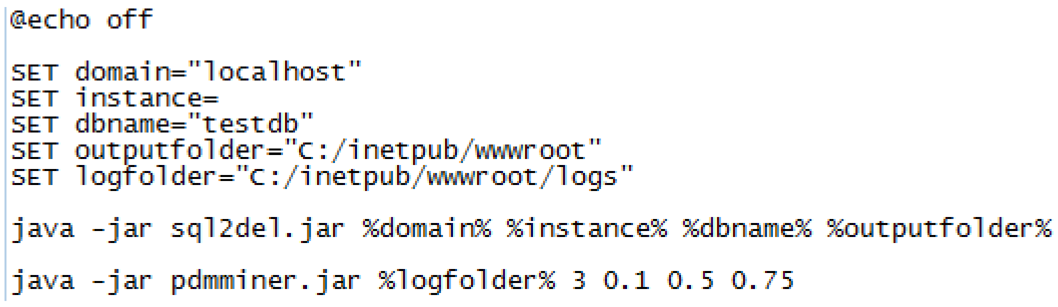
- pdm: containing the resulting model(s) in PDM format
- models: containing an image (PNG) of (each of) the model(s)

The PDM files can be used to visualize the models using the GUI of ProM. Furthermore, if/when more plug-ins become available in the future, these files might be used in order to do further analysis using those plug-ins.

So the total output that is generated consists of three files (per dossier type):

- XES file: containing the document log used for mining
- PDM file: containing the PDM that was resulted from the mining process
- PNG file: containing the visual representation of the PDM that was mined

Figure 30 shows the script that can be used to run both JAR files using the parameters that were mentioned above. In this case 3 models will be created using thresholds 0.1, 0.5 and 0.75.



```
@echo off

SET domain="localhost"
SET instance=
SET dbname="testdb"
SET outputfolder="C:/inetpub/wwwroot"
SET logfolder="C:/inetpub/wwwroot/logs"

java -jar sql2del.jar %domain% %instance% %dbname% %outputfolder%

java -jar pdmminer.jar %logfolder% 3 0.1 0.5 0.75
```

Figure 30 Script to run the plug-in

One final remark: Since everything is done through the command line interface and because using all thresholds might be somewhat difficult to understand, the dependency threshold is the only threshold that can be passed as a parameter. In this case, the all-tasks connected threshold is not used and the positive observations threshold has been dependant on the number of traces, the number of events and the number of total instances. Several values for the positive observations threshold have been tested and the following value gave the best results:

$$Positive\ Observations\ Threshold\ (POT) = 0.5 * \left(\frac{Nr\ of\ traces\ *\ Nr\ of\ elements}{Total\ nr\ of\ instances}\right)$$

6.4.3 Displaying the models

A webpage is made in order to display the different number of images created by the mining plug-in for each dossier type. This webpage is made available/accessible to users of Serenga through a link in Serenga. Following this link, the PNG images for the current dossier type are displayed so that users can see the model of the process (dossier type) they are currently working on.

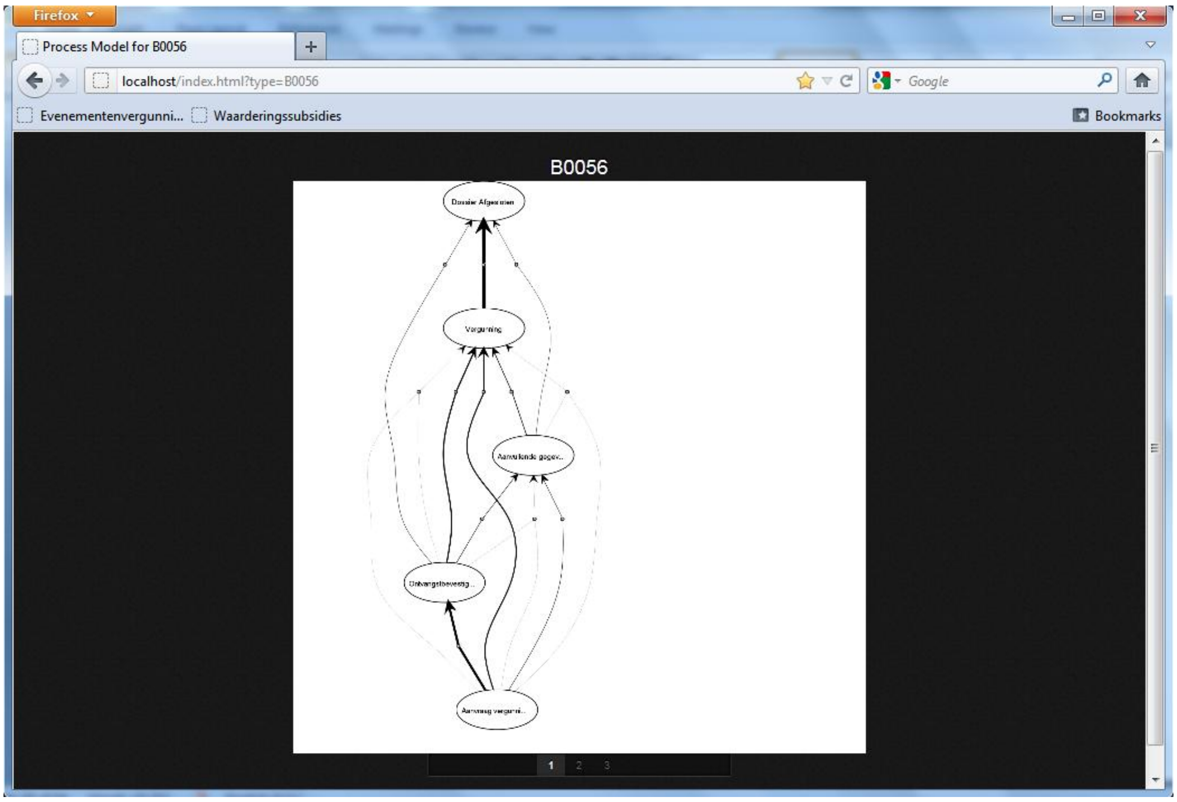


Figure 31 Webpage showing a PDM

In Serenga a link can be created using the %%processid%% variable. This variable can be used to specify the ID of the Dossier Type that is currently being worked on. The URL that is used to access the webpage has the following structure:

*http://**domain**/index.html?type=**%%processid%%***

A screenshot of the webpage is shown in Figure 31 (in which case the webpage is hosted locally and the processid = B0056).

6.5 Evaluation measure

In process mining, *conformance checking* is used to relate a model to an event log and vice versa. Conformance checking indicates to what extent the behavior described by a model and the behavior as recorded in a log are similar and to what extent they differ from one another. As such, conformance checking is used to measure the performance of process discovery algorithms.

This section describes the evaluation of the PDM Miner. In section 6.5.1 a simple calculation is used to indicate the number of traces that can be successfully executed by the PDM created by the mining plug-in. This measure is used in section 6.6 to indicate the performance of the PDM Miner on actual usage data provided by Serenga. Section 6.5.2 explains why the common measure, the *fitness* measure cannot be used in this case.

6.5.1 Successful Executions

The measure that is used for this project is the successful executions (SE) measure. This measure uses a PDM that is created by the mining process and tries to execute each trace in this model by replaying all traces using the model. This measure uses the ‘all or nothing’ principle as a trace can either be (successfully) executed completely or it can fail (in contrast to the fitness measure described in the next section).

The total number of successful executions is the sum of the successful executions of all individual traces. Figure 33 illustrates how a trace is executed with the use of a PDM, for this the mortgage PDM shown earlier is used.

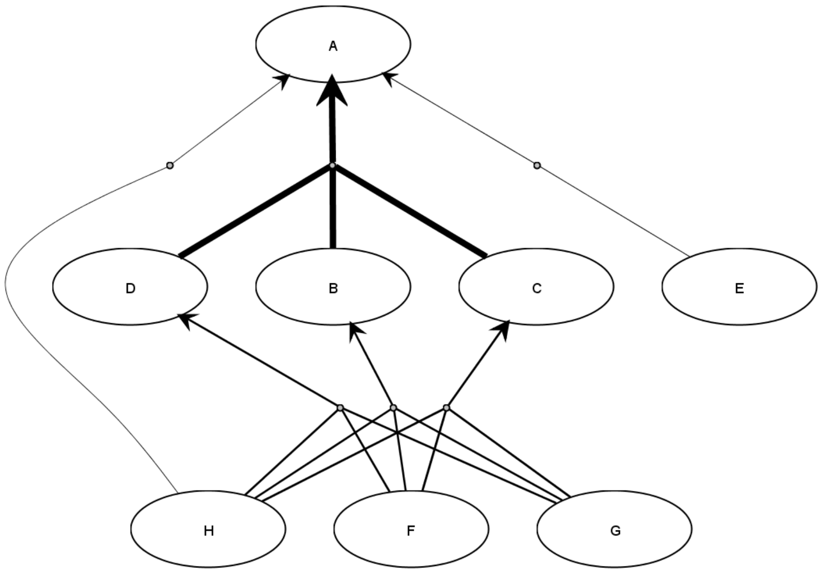


Figure 32 Mortgage example created with the PDM Miner

As an example, we use the trace GHFBCDA to show how this trace is executed using the PDM shown in Figure 32. Figure 33a shows the nodes that are initially enabled. These are the nodes that are the output of an operation that has no input documents. These nodes are represented by a yellow ellipse indicating that these nodes are enabled. After this, we retrieve the first document in the trace, which is document G. In Figure 33b the node for document G is colored green, indicating that this document has been processed. In Figure 33c document H has been executed, which enables document A. The next document in the trace is document F. After this document has been processed, document C becomes enabled (shown in Figure 33d) since the operation that has document C as its output now has processed all its input documents and hence the operation can be executed. This process is repeated for the documents B, C and D, as is shown in Figure 33e, after which document A would become enabled, but A was already enabled when H was executed. Finally, the last document of the trace is processed and hence this trace is successfully executed using the PDM.

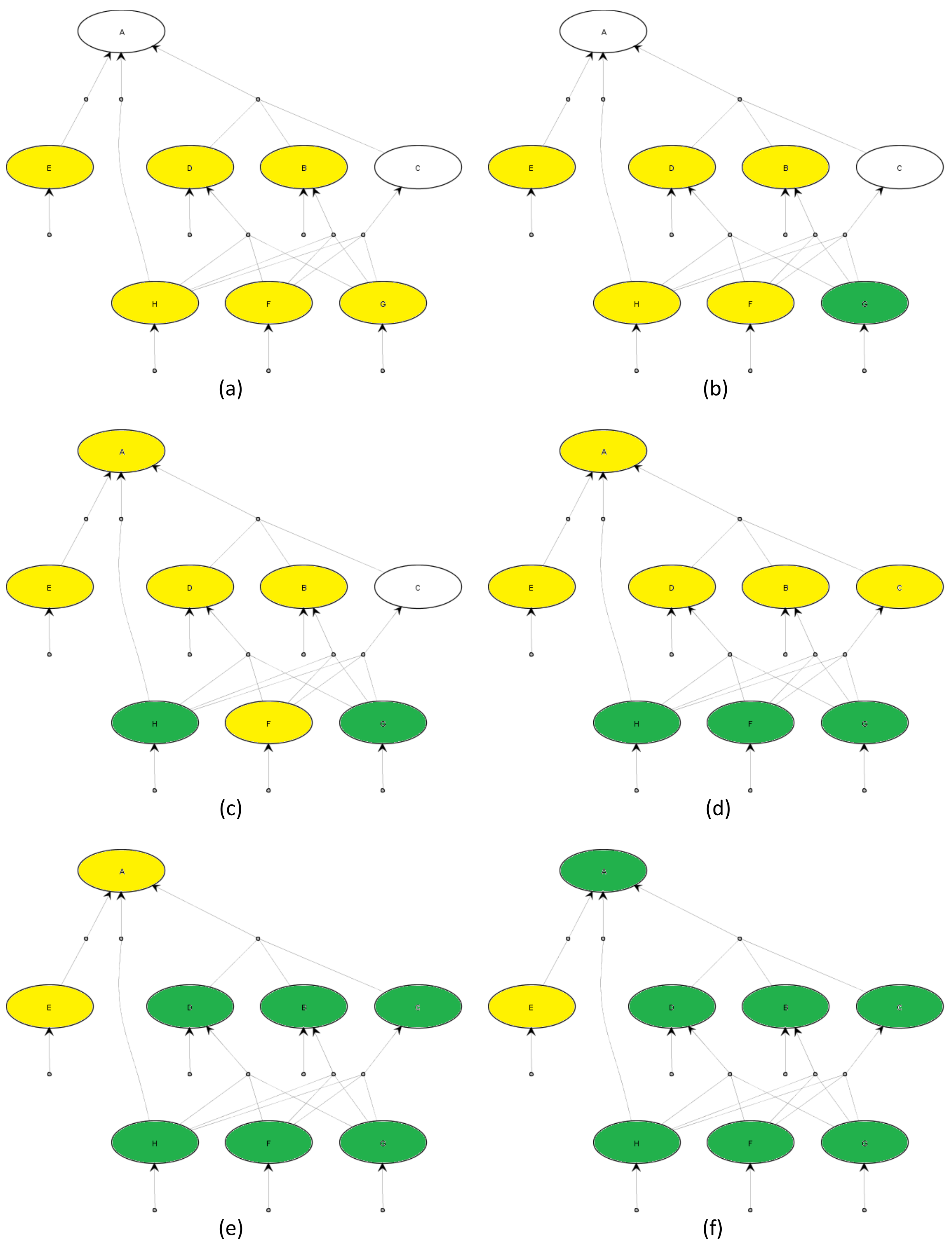


Figure 33 Successful execution of trace GHFBCDA using enabled (yellow) and already executed (green) nodes

If at some point during the execution of a trace a certain document is not enabled or the document has already been executed before, this means that the trace cannot be further executed and hence the execution of the trace fails.

This process is repeated for each trace that is present in the document log and, as a result, returns the total amount of traces that can be executed (and the total amount that cannot be executed). This value is then expressed as the percentage of traces that can be executed successfully. Therefore, the performance of the PDM Miner can be measured by how well the resulting model represents the traces the model was based on.

The successful execution measure of a trace σ by a PDM M is therefore defined as:

$$SE(\sigma, M) = \begin{cases} 0, & \text{if } \sigma \text{ can be replayed entirely by } M \\ 1, & \text{if } \sigma \text{ cannot be replayed by } M \end{cases}$$

Hence, the successful executions measure of a document log L by a PDM M is defined as:

$$SE(L, M) = \frac{\sum_{\sigma \in L} SE(\sigma, M)}{|L|}$$

Where |L| is the number of traces in the document log L.

However, a model that has a high SE percentage does not necessarily make it a good model in the sense that it understandable. The model with the highest SE percentage is also the model containing the most information, which in most cases also makes the model less clear to understand. As we will discuss in chapter 7, other factors might be (more) important when deciding whether or not a model is a good model.

6.5.2 Fitness

Fitness is a measure that is commonly used to indicate how well a model represents the traces that it was based on. Instead of just measuring how many traces can be executed successfully or not, the fitness measure indicates how much of a trace can be executed by the model. For this, each trace is replayed using the model.

In [12] fitness of a single case with trace σ on a net N is defined as:

$$fitness(\sigma, N) = \frac{1}{2}\left(1 - \frac{m}{c}\right) + \frac{1}{2}\left(1 - \frac{r}{p}\right)$$

Where p is the number of tokens that is produced, c is the number of tokens that is consumed, m is the number of missing tokens and r is the number of remaining tokens. In general, when a task is executed, it consumes all the tokens in its input places and produces a token in each of its output places. If a trace can be replayed entirely, this means that whenever a task appears in the log, its input places contain a token in the model. If this is not the case, for each place that does not contain a token, the missing tokens count is increased. When the entire trace has been replayed and there are any token left (besides in the final/terminating place), these are counted as remaining tokens.

Fitness of an event log L on a net N is hence defined as:

$$fitness(L, N) = \frac{1}{2}\left(1 - \frac{\sum_{\sigma \in L} L(\sigma)\ x\ m_{N,\sigma}}{\sum_{\sigma \in L} L(\sigma)\ x\ c_{N,\sigma}}\right) + \frac{1}{2}\left(1 - \frac{\sum_{\sigma \in L} L(\sigma)\ x\ r_{N,\sigma}}{\sum_{\sigma \in L} L(\sigma)\ x\ p_{N,\sigma}}\right)$$

However, fitness is only useful when the net or model that it is applied to has a clear execution semantic. In a PDM fitness cannot be used to give an indication of how much of a trace can be executed by the model.

In a PDM, whenever an operation is executed, zero or more tokens are consumed and exactly one token is produced. There is however a problem with calculating missing tokens; in contrast to traditional process mining, where the activities (events) are the main elements, in our case the main elements in a log are the documents (and not the operations on these documents). This leads to the following problem: If there is an XOR routing (i.e. a document that can be created by two or more operations) in the model, for example the one shown in Figure 34, there is no way to determine which operation caused the creation of document X. Furthermore, when none of the documents needed to create document X are available (A, B, C have not been seen (yet) in the current trace), it is not possible to determine whether we should increase the missing counter by 1 or 2, since we don't know which documents (and which operation) have lead to the creation of document X in the trace.

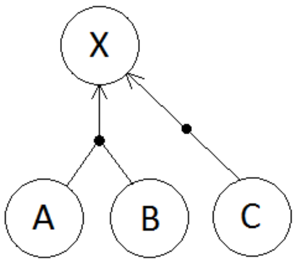


Figure 34 Element X is output of two operations (XOR)

6.6 Dutch Municipality

The data that is available on the processes of the municipality contains 195 different dossier types. Each dossier type has a different number of dossiers (traces) within this data, ranging from only 1 to 399. Several dossier types are chosen based on the number of traces to be able to test the algorithm with different amounts of available data. The number of document groups (elements) per dossier type ranges from 1 to 39.

The process with the most traces, or dossiers, is the 'Evenementenvergunningen' process or 'Eventpermits' in English and this process has 399 traces. This process has 11 different document groups.

The figures on the next pages show the three different PDMs that are generated when running the PDM Miner through the command line interface using the three different thresholds of 0.1, 0.5 and 0.75 respectively.

Figure 35 shows the PDM that was created for a dependency threshold of 0.1. This PDM has the most detail as only few dependency relations are filtered out. Figure 36 has more dependency relations that are ignored due to the higher dependency threshold of 0.5 and hence, the model becomes more comprehensible. Finally, Figure 37 removes some more (uncommon) behavior and, with a dependency threshold of 0.75 still gives a good representation of the process.

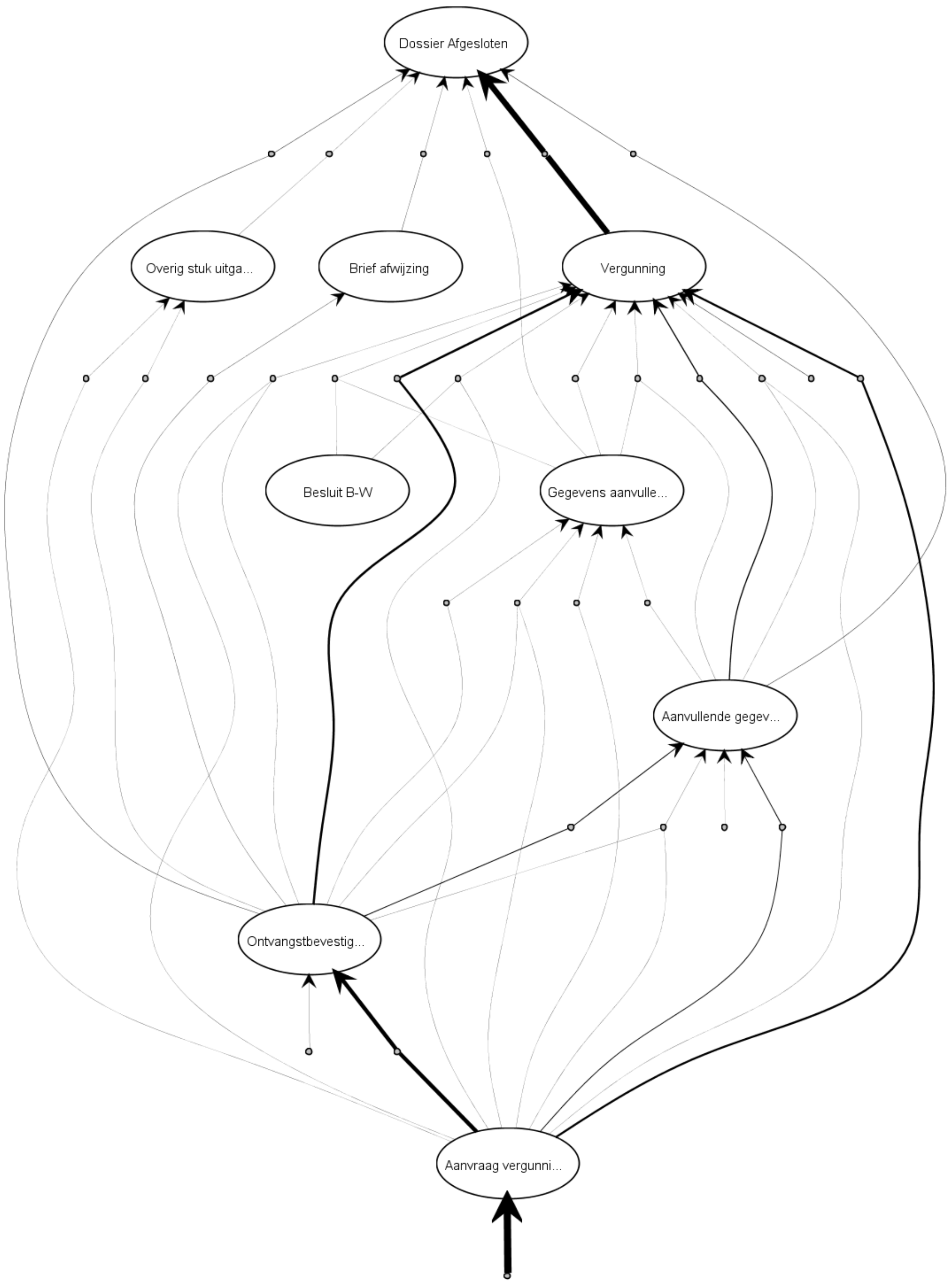


Figure 35 'Evenementenvergunningen' process with dependency threshold set to 0.1

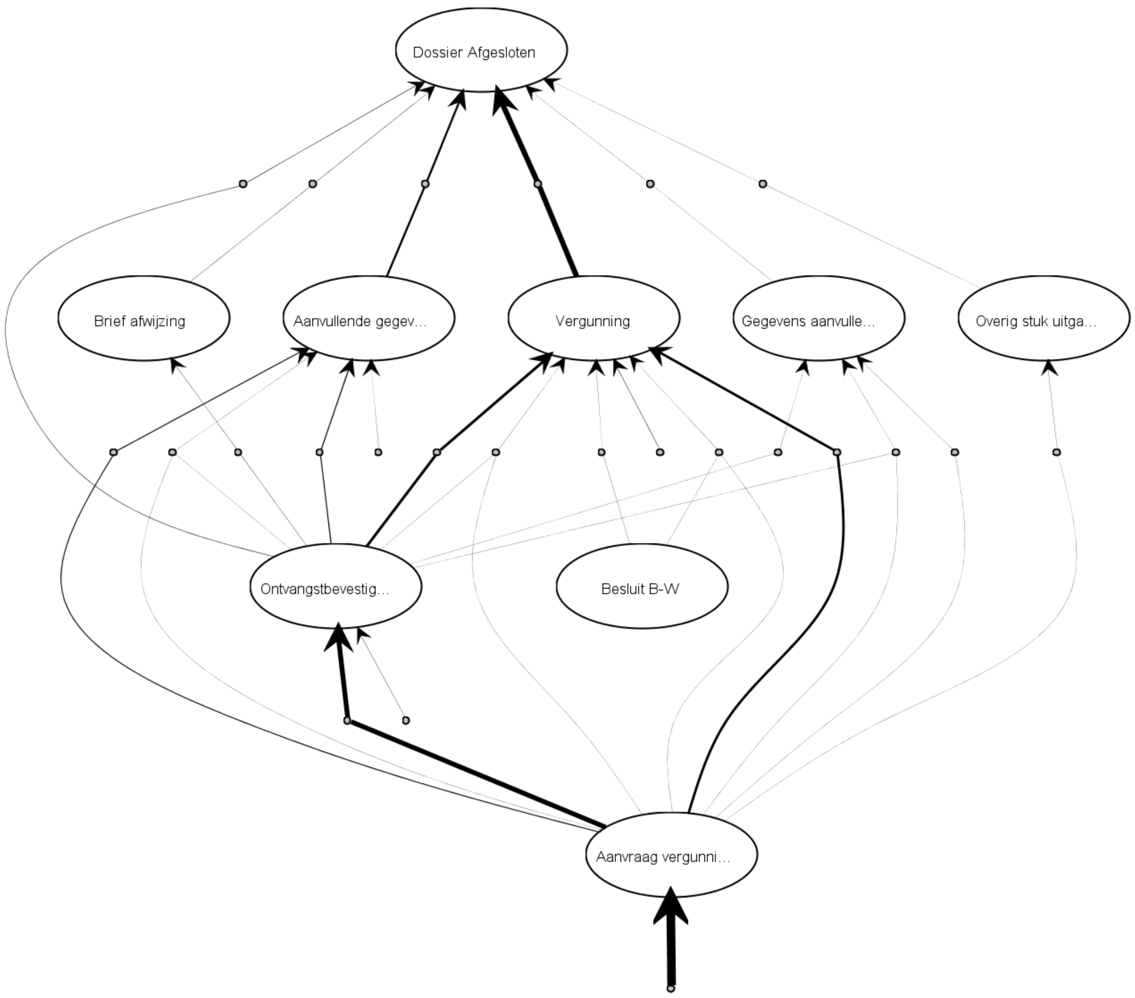


Figure 36 'Evenementenvergunningen' process with dependency threshold set to 0.5

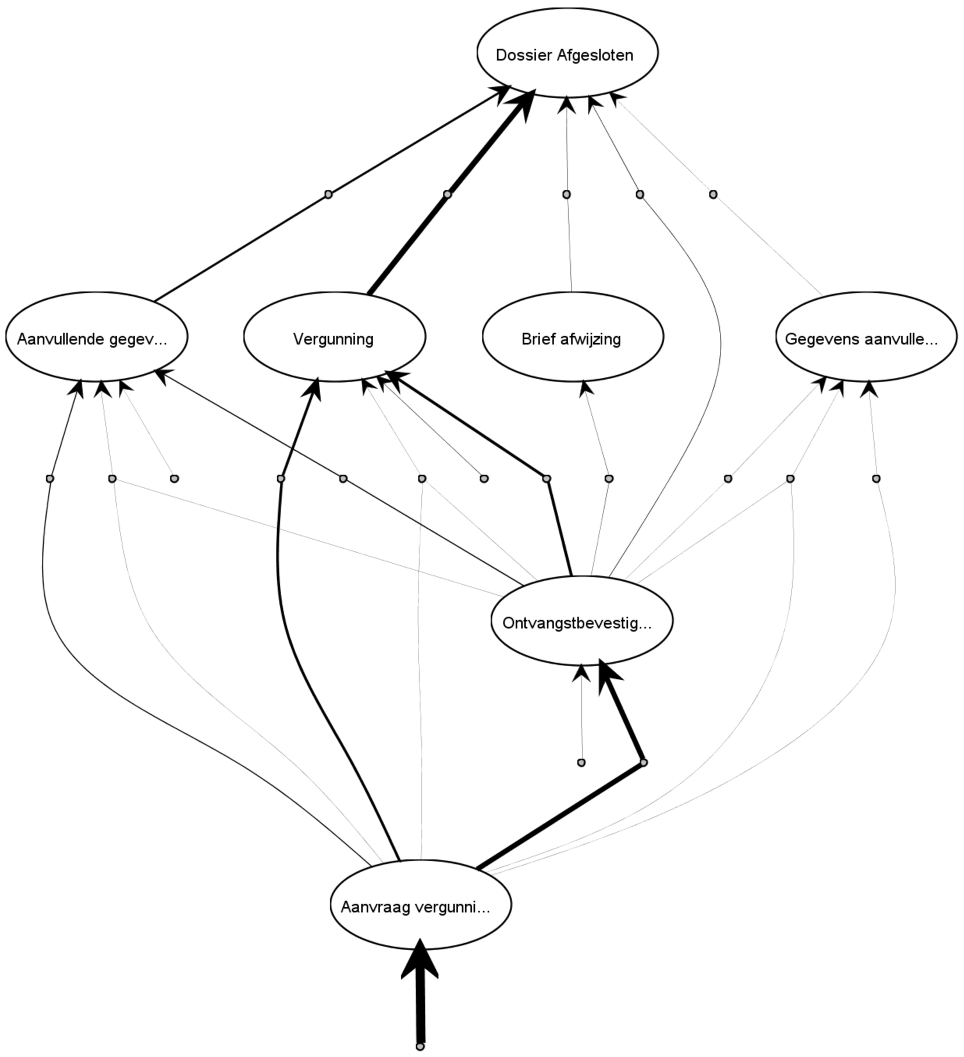


Figure 37 'Evenementenvergunningen' process with dependency threshold set to 0.75

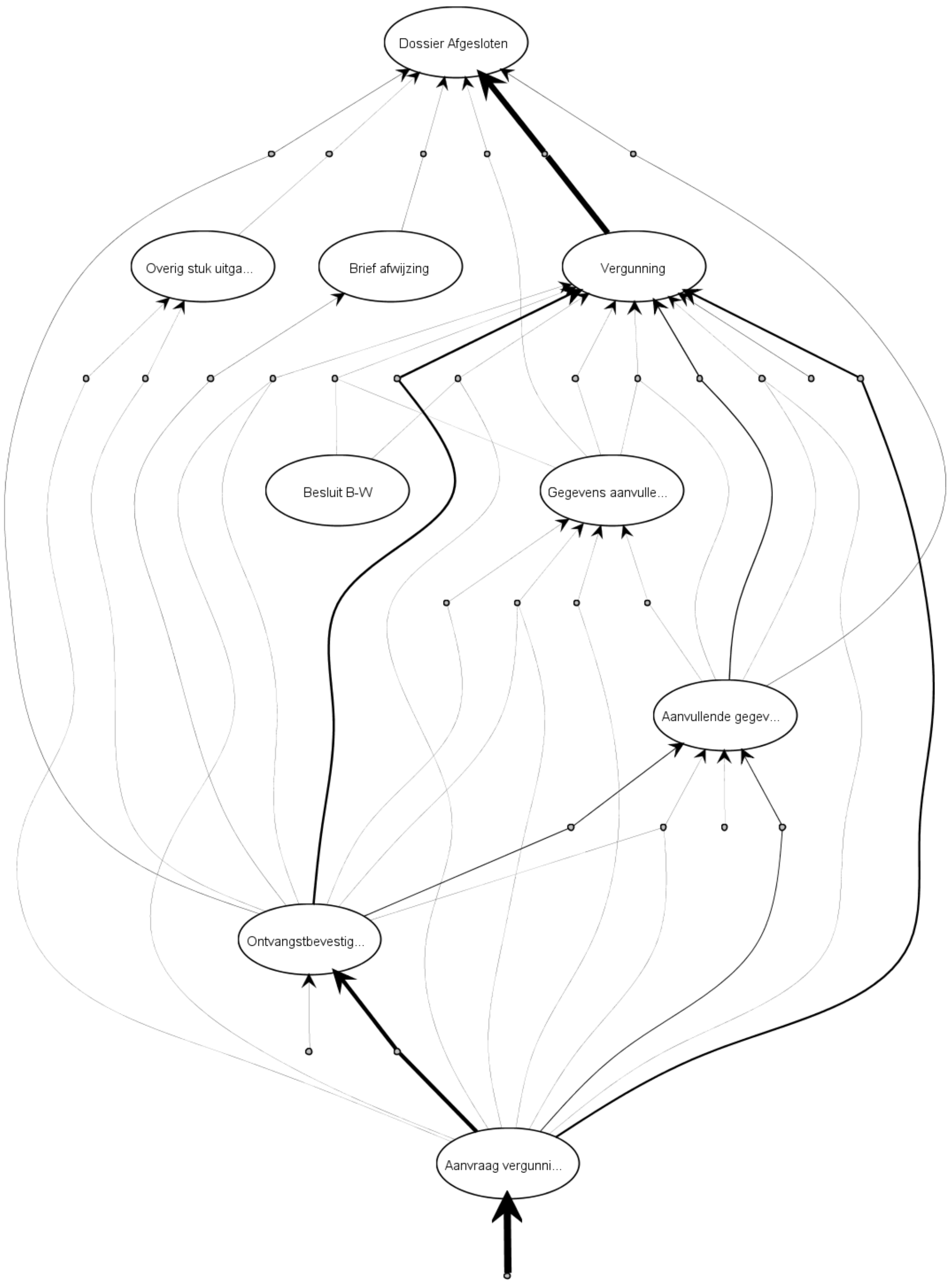


Figure 35 'Evenementenvergunningen' process with dependency threshold set to 0.1

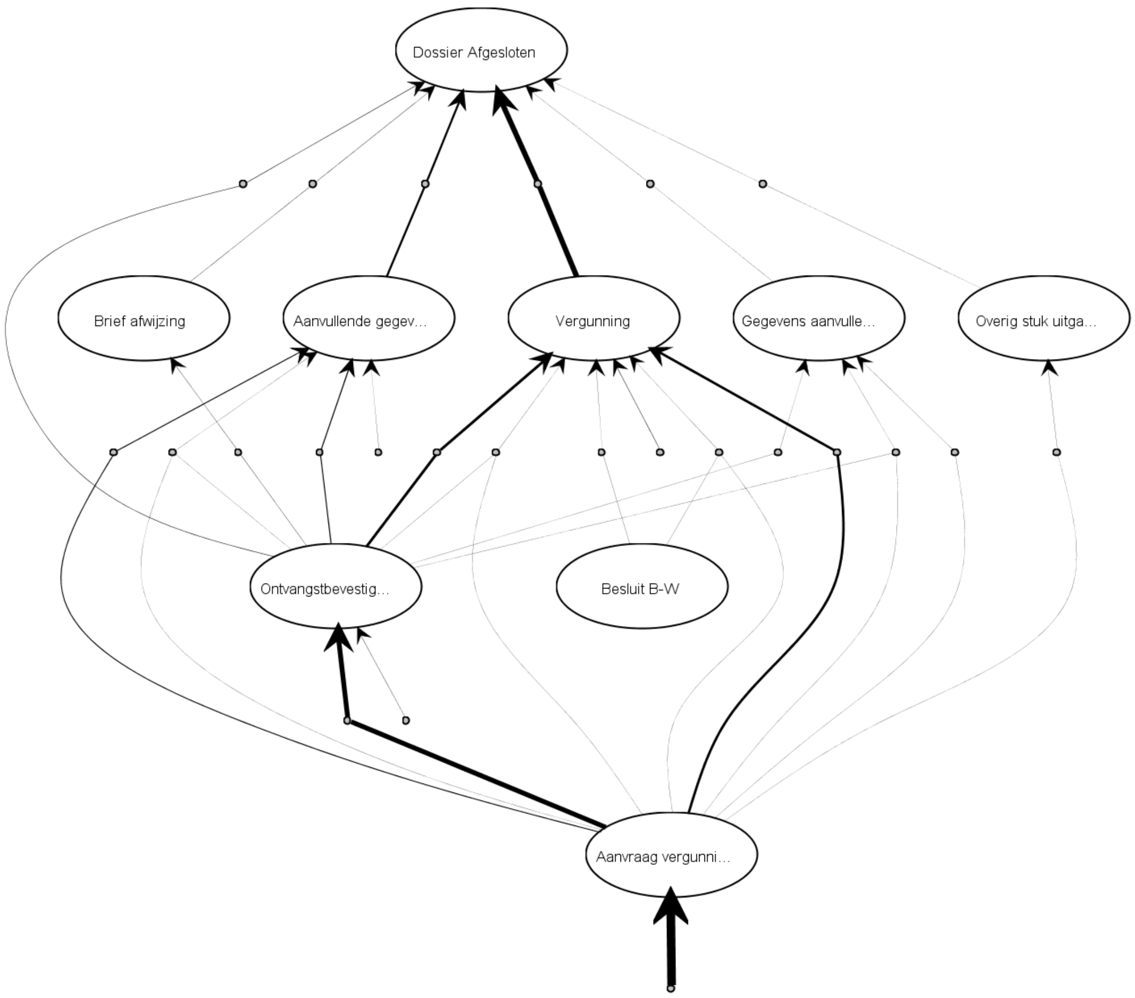


Figure 36 'Evenementenvergunningen' process with dependency threshold set to 0.5

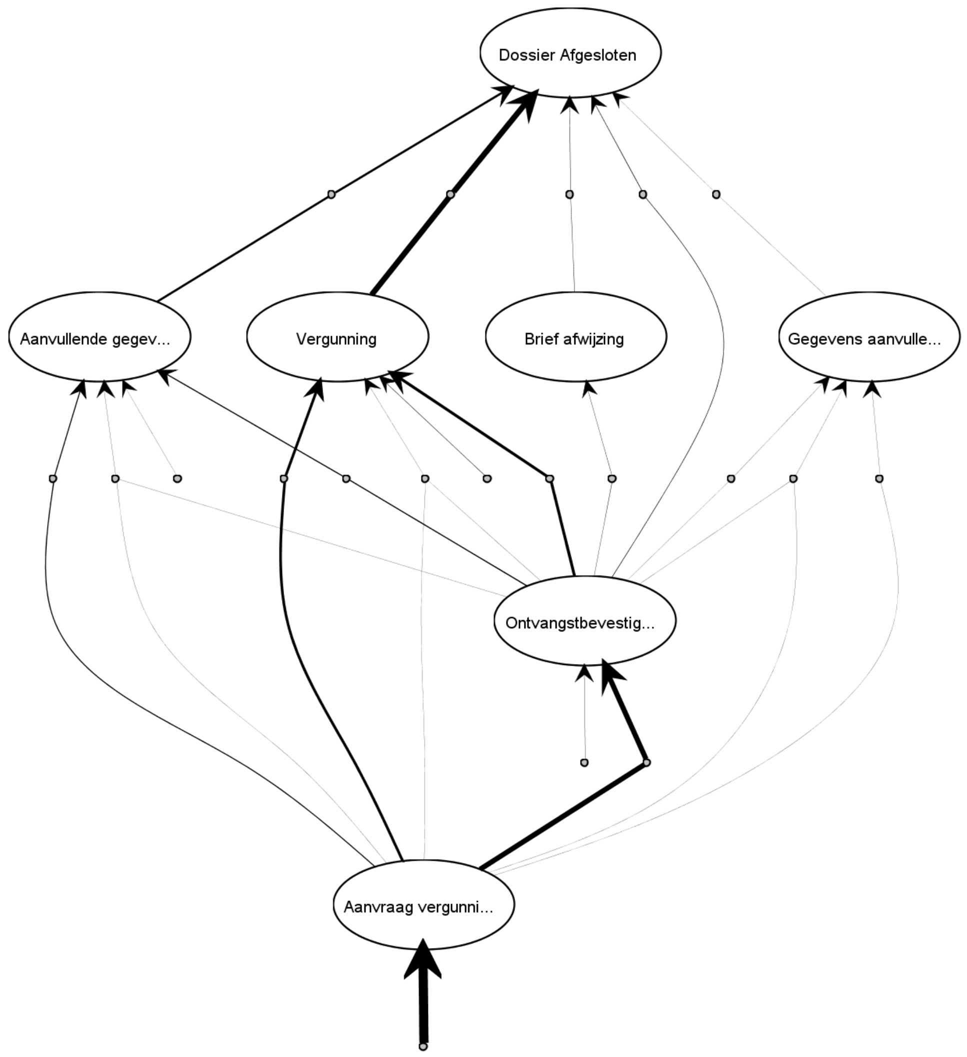


Figure 37 'Evenementenvergunningen' process with dependency threshold set to 0.75

The figures show that there is one path that is taken considerably more often than the other paths. According to these models, the most common way to handle a permit request is to first send a letter confirming that the request was received after which a permit is created and hence the request is granted.

Calculating the number of successful executions when replaying the log shows that with for the first two dependency measures (0.1 and 0.5), out of the total of 399 cases, 6 cases cannot be (fully) executed. The other 393 traces can be successfully executed and completed resulting in a successful execution percentage of 98.50% for all three models. Using the threshold of 0.75, results in 17 failed executions versus 382 successful executions, which equals 95.74% of the traces.

In the remainder of this section we give 4 examples of processes that are used to mine PDMs. These processes are chosen based on the number of traces (dossiers) and the number of different document groups that are used. This distinction is made because the more times a process is executed (more traces), the easier it should be to detect common behavior (assuming there is any). On the other hand, the more document groups are used for a process, the more complex the models tend to get. Table 13 shows the processes that are selected and indicates why they were chosen. The first column states the name of the process (dossier type). The second and third columns indicate the amount of traces/elements with respect to the other processes. Column four and five give the exact number of distinct traces/elements of each of the processes. The last column gives the total number of instances (elements in the whole document log). This number divided by the total number of traces indicates the average number of elements per trace.

Dossier Type	Amount of Dossiers	Amount of Document Groups	Nr. of Dossiers	Nr. of Document Groups	Nr. of Instances
Waarderingssubsidies	Many	Many	332	31	1398
Zienswijzen	Many	Few	361	4	799
Investeringssubsidies	Few	Many	35	39	187
Gemeentelijk proces	Few	Few	1	4	4

Table 13 Selected processes

Process 1: Waarderingssubsidies

The first process is called 'Waarderingssubsidies'[6] in Dutch. This process is shown because it has 332 dossiers and 31 different document groups which is high in comparison with the other processes. Hence, it is a process with a high number of traces and a high number of document groups.

Figure 38 shows the PDM that is created after running the algorithm with a dependency threshold of 0.75. For the other two PDMs created with dependency thresholds 0.1 and 0.5, see Appendix A. The first thing to notice in the PDMs that are created is that there are far fewer elements than the 31 document groups that were mentioned in Table 13. This is due to the fact that although there are 31 different document groups, not all these document groups appear a significant number of times in the traces as can be seen in the direct succession matrix (and the dependency measures matrix) shown in Appendix A.

[6] A 'waarderingssubsidie' is a subsidy given as a token of appreciation or encouragement to execute or organize certain activities

The successful executions measures of these three models are 91.27%, 87.05% and 68.67% respectively. When looking at the models, it shows that although the first model with a dependency threshold of 0.1 has the highest number of successful executions, it is also the model with the most arcs making it harder to understand. On the other hand, the last model with a dependency threshold of 0.75 is the most understandable/readable of the three, but it also has the lowest number of successful executions.

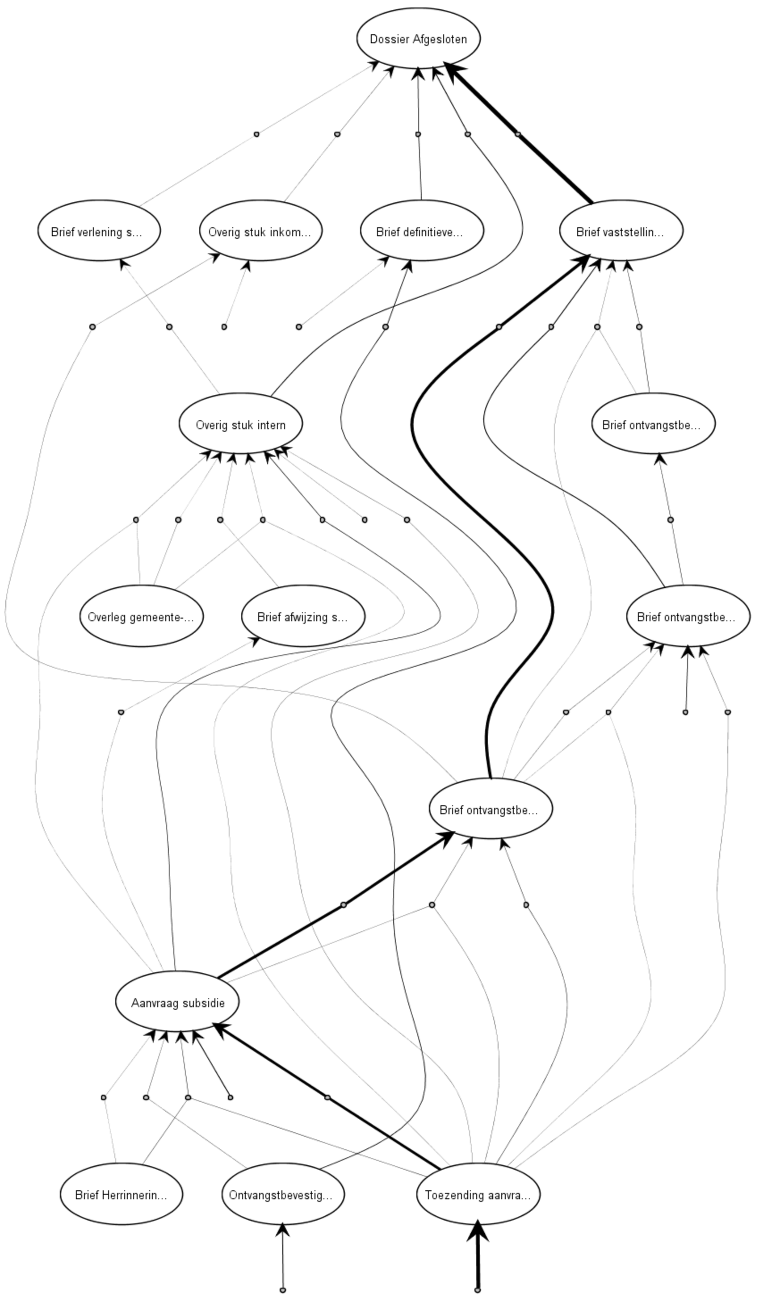


Figure 38 PDM for the 'Waarderingssubsidies' process with dependency threshold set to 0.75

Process 2: Zienswijzen

The second example process we will show is the 'Zienswijzen'[7] process. This process has many traces, but only has 4 different document groups, with an average of 2.2 document groups used per process.

The PDMs that are created by the PDM Miner plug-in are shown in Figure 39. In this case the first two models, created for the thresholds 0.1 and 0.5 are identical (besides the layout).

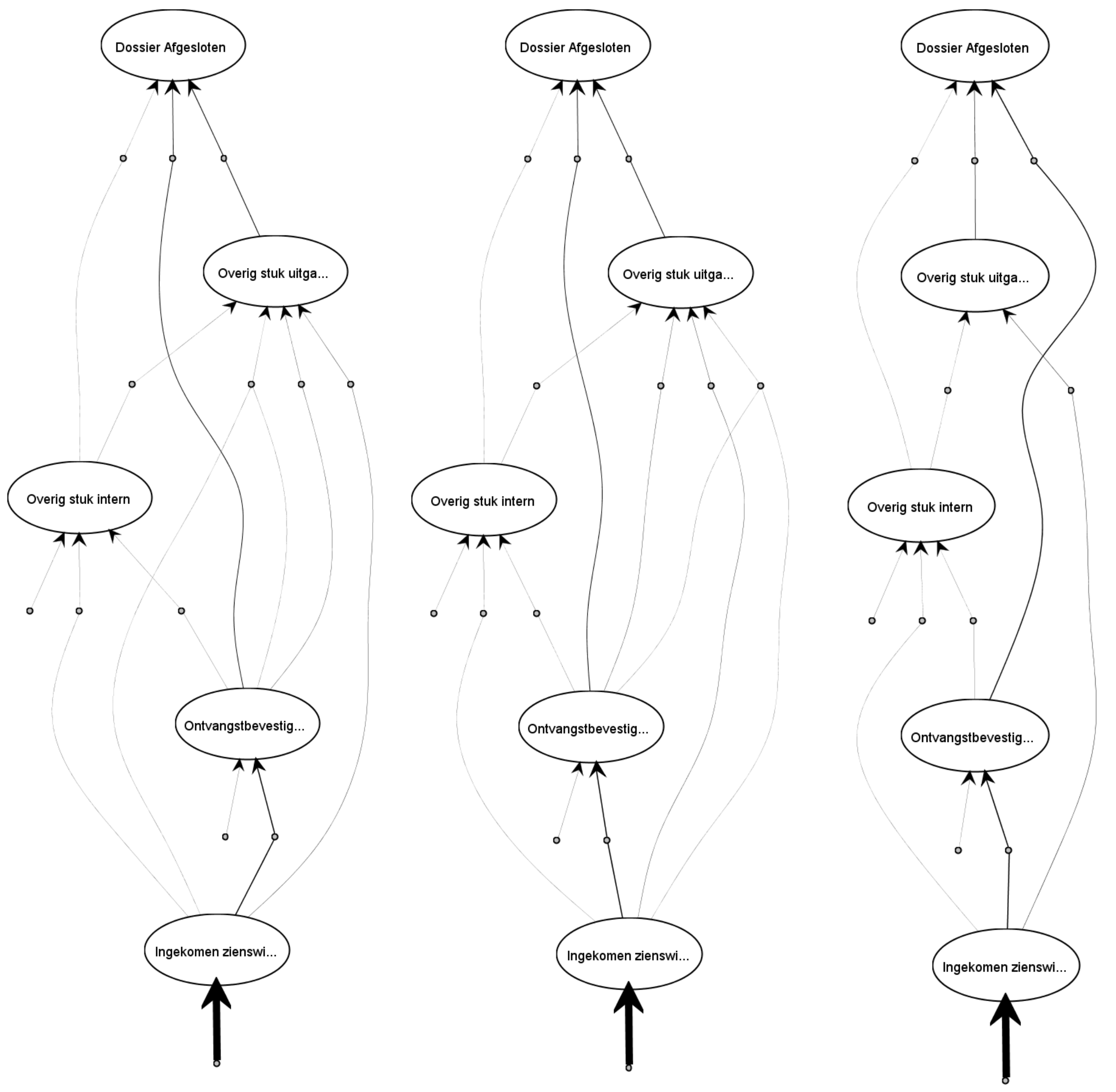


Figure 39 PDM for the 'Zienswijzen' process with dependency threshold of 0.1, 0.5 and 0.75

The number of successful executions is in this case again identical for all three models. For all three models, all traces can be successfully executed resulting.

[7] A 'zienswijze' is an opinion on an intended decision made by a municipality

Process 3: Investeringssubsidies

The next process is 'Investeringssubsidies'[8]. The document log of this process only contains 35 traces, but on the other hand is also the process with the most document groups, 39. The total number of instances in this process is 187. This implies that the average trace consists of more than 5 elements.

Figure 40 shows the model that is created for a dependency threshold of 0.1. Although this model looks nice, only 28.57% of the traces can be successfully executed using the model. When looking at the numbers, this can be explained by the fact that there are only 35 traces. Moreover, there are 187 instances, which implies an average of around 5 elements per trace. But, there are a total of 39 different elements used throughout the traces. Hence, a lot of elements only occur once or twice in the log. These elements are in this case filtered out since the positive observations threshold is set to 3.

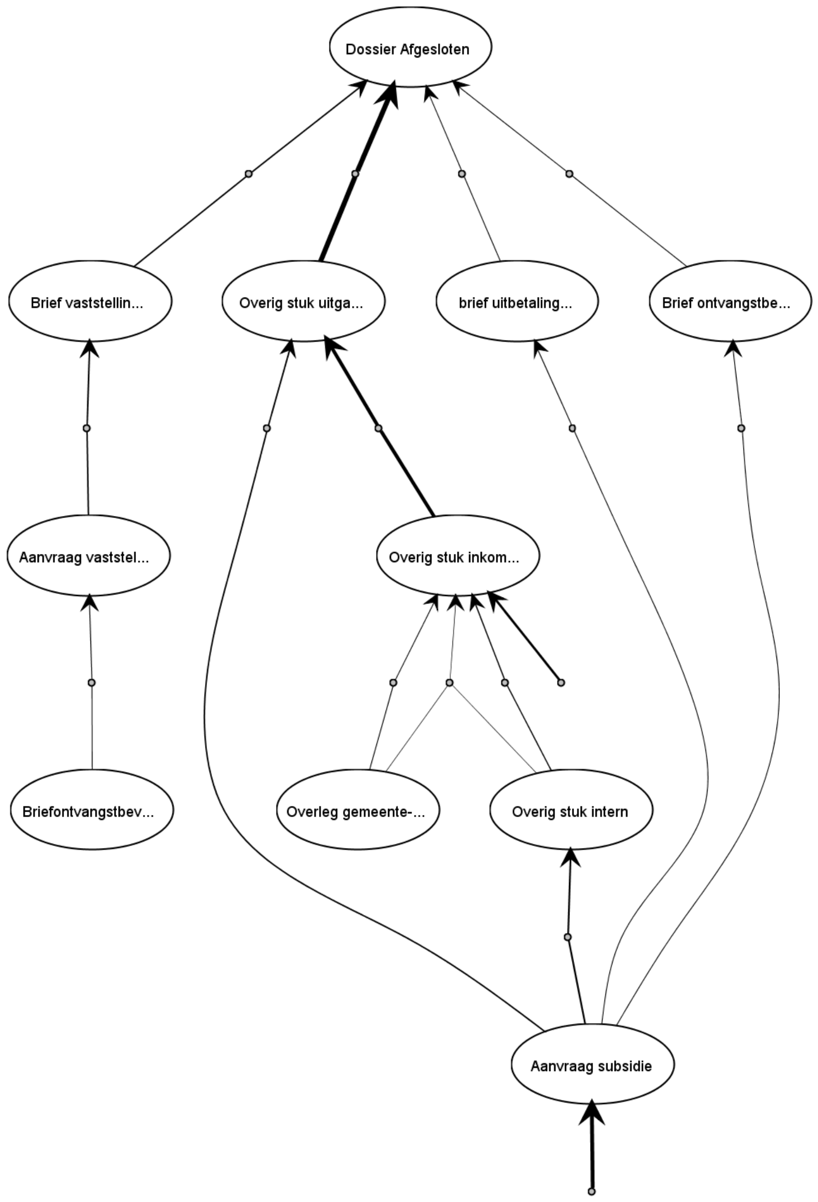


Figure 40 Model for the 'Investeringssubsidies' process with dependency threshold of 0.1

[8] An 'investeringssubsidie' is an investment of a municipality in property or real-estate

The last process we show here is a process that has only one trace. This process is called ‘Gemeentelijk proces’ or municipality process and consists of 4 different document groups. The resulting PDM for the thresholds 0.1 and 0.5 is shown in Figure 41.

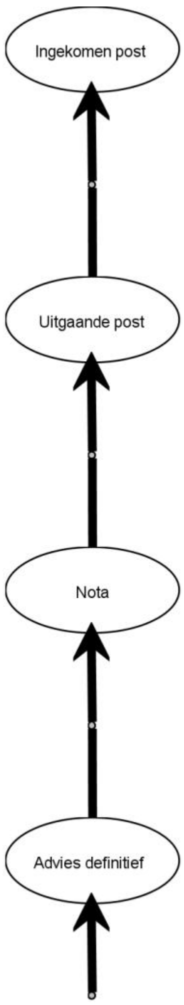


Figure 41 PDM generated for 'Gemeentelijk proces'

Since there is only one trace in this last process, the PDM shows a sequence of document groups. This is due to the fact that there is no way to determine the joins. For the third threshold, with a value of 0.75, the PDM that is created for this process only consists of the (artificial root) element. This is because all dependency measures are below the threshold. Hence, this also results in the first two models being able to successfully execute 100% of the traces where the third model is obviously not able to successfully execute any of the traces.

Recall the definition of the dependency measure in section 5.3:

$$a \Rightarrow b = \left(\frac{|a>b|-|b>a|}{|a>b|+|b>a|+1}\right) \text{ if } (a \neq b)$$

Hence, if |a>b| = 1 then |b>a| = 0 since in one trace, each element can only occur once. This results in the following dependency measures: $a \Rightarrow b = \frac{1}{2}$ and $b \Rightarrow a = -\frac{1}{2}$. Since we only have one trace, this goes for all dependency measures; all dependency measures are equal to either -0.5, 0 or 0.5.

Since the dependency measure to generate the third model is set at 0.75, all dependency measures automatically are below this threshold. To illustrate this, none of the original elements is displayed in the model, but the artificial root element is displayed instead.

6.7 Conclusions

This chapter first showed what data was available from Serenga and what data was best to use in order to create a document log. Then we showed how to generate the document logs by getting the data from the Serenga data base. For each dossier type in Serenga a document log is created which can be used as input for the PDM Miner plug-in. In order to generate models using the PDM Miner plug-in an example was given in which three different dependency thresholds were used.

Section 6.5 introduced a measure, the successful executions measure, to indicate how well the PDM Miner performs. This measure counts the number of traces that can be successfully executed by the generated model. However, maximizing the number of successful executions might not necessarily result in the most useful models. A model that is able to replay a high number of traces might contain (a lot) of uncommon behavior, hence the use of different thresholds to filter out certain behavior. On the other hand, filtering out this uncommon behavior automatically lowers the number of successful trace executions as the traces containing this uncommon behavior can no longer be executed by the model. The number of successful executions can therefore give an indication of how much of the total behavior is represented by a model, but it does not automatically say which model is the better one when comparing two models. Which model is better depends on the purpose for which the models are created.

Section 6.6 showed some examples were the PDM Miner was applied to cases of a Dutch municipality that were handled by Serenga. From these examples it can be concluded that it is difficult to get meaningful and readable models. Especially when limiting the interaction between the user and the plug-in. Using preset thresholds for a (large) number of processes can result in reasonable models, however, in order to the best possible model for each of the processes one would have to look at these processes individually and adjust the thresholds accordingly.

7 Discussion

During this project a workshop session was held with experts on PBWD and PDM modeling and the people from Océ that are working on Serenga. The feedback that was collected from this session is discussed in this section. This feedback can be used as further validation of the models that are created using the PDM Miner plug-in described in this report.

7.1 Workshop session

During the workshop session a couple of points where brought forward. This section gives an overview and discusses each of these points briefly.

Exceptional behavior

In some cases it would be useful to have a model containing only exceptional behavior.

Although it goes against the general idea of process mining to get a general view of a process, in some cases, there might also be the need to model the exceptional or uncommon behavior instead of the most common behavior. When a model is used to derive recommendations and guide a user through a process the user is currently working on, it might especially be useful to provide these recommendations in exceptional cases since in that case the user might not be familiar with the way these cases should be processed. In traditional process mining, uncommon behavior is in most cases not relevant, however, in document-based processes these uncommon cases might just be the interesting ones.

As the word 'exceptional' indicates, this behavior might only be observed in a few traces. This makes it difficult to create a model that represents the common way in which exceptional cases were handled, i.e. it might be difficult to give meaning to input/output relations that are derived. There is however also another reason why modeling this exceptional behavior is a difficult task; in many cases, only some parts of a process are exceptional. Moreover, a distinction between exceptional behavior and noise cannot be made; both situations occur only in a (very) small number of cases.

There are three options that can be taken when one does want to model exceptional behavior:

- Model only the exceptional behavior
- Model all possible (common and exceptional) behavior
- Model exceptional behavior and some common behavior

The first option would be to use a threshold to select exceptional behavior and only model this behavior. However, this will almost always result in a disconnected graph consisting of (many) small clusters representing the exceptional parts of the process. Other than showing the exceptional parts of the process, a model created in this way will not be very useful because parts of the process might be missing.

Another option would be to use no thresholds and just model everything. As mentioned before, this will (for most processes) result in a spaghetti-model which is not very useful either. However, with such a model it would be possible to provide recommendations more easily.

The third option would be to use the exceptional behavior and combine this with common behavior to get a connected graph. Probably the best option in this case would be to first mine the log for these exceptional cases (cases which contain exceptional behavior) and then use only the traces for these cases to construct a model. In this way, the exceptional behavior will be captured in the resulting model and the graph will be connected.

In our situation we chose to focus on getting a general model. What we did do however was indicate what behavior in such model is common and what behavior is uncommon by using different thicknesses for the arcs. In this way it is possible to still show some uncommon behavior in the model without losing sight of the general behavior.

Recommendations

In the plug-in created for this project, we use the entire history of a dossier type to create a model which in turn is used to indicate the common and uncommon paths used to complete a certain process. Instead of using the generic model to provide the user with recommendations, the information in the document log can be used to provide recommendations. Given the current state of a dossier, (partial) traces that are similar to the steps taken up to the current point can be used to calculate the step that was taken next in most previous cases. This will in general give more case-specific recommendations; however, chances are that in some (new) cases, no recommendation can be given. If up to a certain point a process was executed in a way similar to previous cases, recommendations based on these historical cases are given. If, from this point, the case deviates from previous cases, no recommendation can be given. Using the generic model one might still be able to provide recommendations looking at all the historical executions of the process.

Simplify thresholds

Not everyone might understand the meaning of all the different thresholds. Therefore, it would be nice if some thresholds could be combined into a single threshold, so that only one value has to be specified for this threshold. However, most thresholds are used in a different context and therefore it would not be possible to combine them. Setting some thresholds to a fixed value however might reduce the thresholds that a user would have to set. However, applying the algorithm to a large number of different dossier types might not give the best possible results using these fixed values.

Container concepts

For this project we used document groups from Serenga usage data as the main elements for the mining process. As mentioned before, these document groups can contain a number of documents which in turn contain the real data that decisions are based on. These document groups could in essence again be the result of a PDM in which the elements are documents. This way, the document groups become container types that can contain a model that represents the document structure within such document group.

As discussed earlier, in our case there was no way to extract useful information ‘smaller’ than the document groups in order to create PDMs of other element.

Life cycle

Instead of assuming a document or document group is available or not (has been added), life cycles could be introduced to indicate the state of a document or document group. In our case a document group could have several states. It could be created, after which documents could be added until eventually the final document is added and a document group becomes ‘complete’ for a particular dossier.

In the current situation we used the last addition of a document to a document group to indicate that a document group is ‘complete’. The timestamp of this last addition was used in the document log. However, in some cases it might be better to take the first addition or even somewhere between the first and last document added. This last situation would however not be possible as for each dossier a document group can have a different number of related documents and, as explained earlier, documents cannot be related to one another across dossiers.

Additional information/models

In addition to the document groups also activities could be used. These would also have to be mined to come up with a generic model. The two PDMs would then have to be linked/merged in order to be able to show the relation between the two. Also resource information could be modeled. This could be done within the PDM (adding resources to operations), or by creating a separate model. When using a separate model, there are probably other models and mining techniques that are more useful for representing these resources.

Furthermore, additional information could be added to the PDM. For example, the number of times an operation occurred in the document log, that is, the number of times a relation between two or more documents or document groups occurred, or how much time it takes to do an operation.

As mentioned before, the number of times an operation is executed is represented by different thicknesses of arcs. Using this kind of visualization over exact number should make it easier/quicker to see common behavior in a model.

7.2 Artifact-Centric Business Process Modeling

The main point that seemed to come forward from the feedback of this workshop session is that when applying process mining to artifact-centric businesses, there is a need for different views of a process.

In traditional mining, the focus is on modeling the common behavior of a certain process and finding bottlenecks in the execution of a process. In artifact-centric business process modeling however, the focus is on what data is required to get to a certain goal. For this, uncommon behavior might be as important as the common way a process is handled, especially when one wants to provide recommendations for future cases.

Another example is the number of times a certain step was executed in previous cases. For managers or board members, the real numbers might be relevant, whereas for someone executing the process, these numbers might not be that relevant and only a visual indication showing the difference between common and uncommon behavior might be enough.

Adding activities or operations to the models created by the mining process should give an even better representation of the process. When the data allows it, one could derive information about when certain paths are taken during a process, based on the values within documents. This information could then be used to better specify routing elements in a data-centric modeling technique.

8 Conclusions

This report describes a first attempt at creating an automated way of generating process models in a data-driven environment. As this report shows, available process mining techniques can be adapted to work in with data-centric processes. We showed this by adapting the FHM algorithm to work in a document-centric environment. The first step is to create a document log that contains the right information that is useful for the mining process. The second step is to mine this document log in order to get a process model. For this project this is done by applying process mining techniques on document logs that contain useful data information in order to create a Product Data Model (PDM).

First we showed how to create a document log in XES format using available usage data stored in a database. The PDM Miner plug-in, based on the FHM algorithm is then applied to this document log in order to generate a Product Data Model based on the common behavior derived from this log.

The PDM Miner plug-in is applied to usage data of a case-handling system called Serenga, created by Océ. Several different example processes from a Dutch municipality have been shown in this report. From these examples we have seen that although it is possible to generate models that correctly represent the information in a document log and do this in a readable and comprehensive manner, it is difficult to find general threshold values that can be applied to a (large) set of processes. In order to get the best possible models for a number of processes, it is still necessary to look at these processes individually and tweak the different thresholds for each of these processes.

The data that was available for this project limited the possibilities with respect to what could be used for the process mining. Where mining document-centric or data-centric processes would really distinguish itself from control-flow mining is when real data values are available. As we will discuss in the next section, this would make things much more interesting and challenging.

8.1 Limitations and Future Work

8.1.1 'Loops'

The algorithm as described in this report has some limitations. The first and biggest limitation is that loops can still occur in the final PDM. Although, these loops are not actual loops, but rather appear as loops since some execution paths overlap. In practice, with the assumptions that were made, although there are loops, all operations are only executed at most once and each document is only created once. When, for example, the log contains only two different traces, ABCD and CABD, this results in the dependency measures shown in Table 14.

	A	B	C	D	End
A	0	2/3	-1/2	0	0
B	-2/3	0	1/2	1/2	0
C	1/2	-1/2	0	1/2	0
D	0	-1/2	-1/2	0	2/3

Table 14 Dependency Measures

Now in the current implementation, if the dependency threshold is chosen to be below 0.5, all (positive) dependency measures will be accepted, resulting in the output bags shown in Table 15.

I	D_C
[1:{C}]	A
[2:{A}]	B
[1:{B}]	C
[1:{B},1:{C}]	D

Table 15 Output bags

As already can be seen from these output bags, there is a loop containing the documents A, B, and C. This becomes even clearer when a PDM is created from these output bags, which is shown in Figure 42.

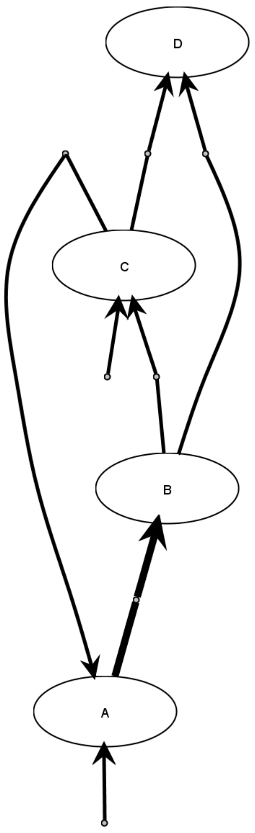


Figure 42 Example of a 'loop'

As the model shows, there is what appears to be a loop containing the documents A, B and C. When looking at the traces (ABCD and CABD) that resulted in the creation of this model however, there are no loops. The loop in the model is the result of the assumption made earlier that a PDM can only contain each element once. Since there are only two traces and the threshold that was chosen was too low to filter anything out this results in the model shown above.

In practice this means that in the PDM it should not be allowed to alter a document that was already created. So in the figure shown above, if the sequence is A>B>C, it is not allowed to execute another operation resulting in one of these documents. In this case the operation with input C and output A would not be allowed to be executed. As stated in [7] operations that potentially lead to the creation of an element that already exists and that already has a value assigned to it, will be skipped. Hence, in the above situation were A, B and C have been created, the only possible operation that can be executed is the operation resulting in the creation of document D.

Although the model shown in Figure 42 is a valid PDM under the assumptions made, a PDM could be created in which no loop exists and which still enables the execution of both traces mentioned above. To obtain such PDM for the traces shown above, the input bags should be as follows:

I	D_C
[]	A
[2:{A}]	B
[]	C
[1:{B,C}]	D

Table 16 Correct input bags

The input bags shown in Table 16 result in a valid PDM that correctly represents both traces.

Note that in the simple case used in this section, the problem could be solved by removing all input relations to documents that are ‘start elements’. Documents A and C both occur at the start of a trace and hence all input relations to these documents would be removed resulting in the input bags shown above, which in turn results in the model shown in Figure 43. However, these loops can also appear in other parts of the model where no start elements are involved.

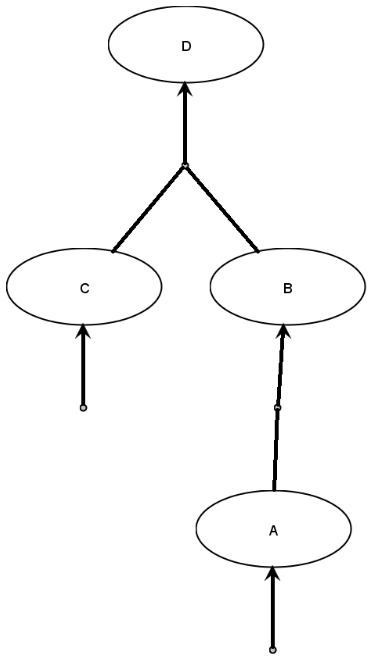


Figure 43 Corrected model without loop

8.1.2 Extending the PDM

The PDMs created by the algorithm for this project can of course be extended to contain more information that can be relevant for a particular process or setting. In the case of Serenga, the system also stores information about activities. An activity in Serenga is basically a hand-over of work to another handler of the dossier. In this situation these activities could possibly be used to refine (some of the) operations.

Furthermore, as mentioned earlier, when real data values are available for the mining process, things become even more interesting. The conditions of operations (or routing elements in general) could then possibly be derived since by looking at the values of data elements (for example data values within documents) it becomes possible to specify when a certain choice is taken during the process. By looking at the values for the associated data elements in a log, we can look at each routing element and determine for which values one choice was taken in the past and for which values another was taken. In order to be able to do this, one could apply clustering/classification techniques on data values using data mining techniques. As a result, recommendations could be provided in a much better way, based on the actual values of data elements. Furthermore, common and exceptional decisions made within processes could be mined and analyzed.

References

[1] A. Nigam and N. Cawell, "Business artifacts: An approach to operational specification," *IBM Systems Journal,* vol. 42, no. 3, pp. 428-445, 2003.

[2] K. Bhattacharya, C. Gerede, R. Hull, R. Liu and J. Su, "Towards Formal Analysis of Artifact-Centric Business Process Models," *BPM,* vol. 4714/2007, pp. 288-304, 2007.

[3] S. Kumaran, R. Liu and Y. Wu, "On the Duality of Information-Centric and Activity-Centric Models of Business Processes," *Lecture Notes in Computer Science,* vol. 5074/2008, pp. 32-47, 2008.

[4] S. Bodker, M. Kyng and K. Schmidt, "Taking the Work out of Workflow: Mechanisms for Document-Centered Collaboration," in *Proceedings of the Sixth European Conference on Computer-Supported Cooperative Work*, Copenhagen, Denmark, 1999.

[5] P. Dourish, W. Edwards, L. A, J. Lampin, K. Petersen, M. Salisbury, D. Terry and T. J, "Extending Documents Management Systems with User-Specific Active Properties," *ACM Transactions on Information Systems (TOIS),* vol. 18, no. 2, 2000.

[6] J. Wang and A. Kumar, "A Framework for Document-Driven Workflow Systems," *Lecture Notes in Computer Science,* vol. 3649/3005, pp. 285-301, 2005.

[7] I. Vanderfeesten, "Product-Based Design and Support of Workflow Processes," PhD Thesis, Eindhoven, 2009.

[8] I. Vanderfeesten, H. Reijers and W. v. d. Aalst, "Product-based Workflow Support," *Information Systems,* vol. 36, no. 2, pp. 517-535, 2011.

[9] W. v. d. Aalst and P. Berens, "Beyond Workflow Management: Product-Driven Case Handling," *International ACM SIGGROUP Conference on Supporting Group Work (GROUP 2001),* pp. 42-51, 2001.

[10] W. v. d. Aalst, M. Weske and D. Grünbauer, "Case Handling: A New Paradigm for Business Process Support," *Data & Knowledge Engineering,* vol. 53, no. 2, 2005.

[11] H. Sharp, Y. Rogers and J. Preece, Interaction Design: Beyond human-computer interaction, Wiley, 2007.

[12] W. v. d. Aalst, Process Mining: Discovery, Conformance and Enhancement of Business Processes, Eindhoven: Springer Verlag, 2011.

[13] C. Günther and W. v. d. Aalst, "Process Mining in Case Handling Systems," *BETA Working Paper Series,* vol. WP 150, 2005.

[14] B. v. Dongen and W. v. d. Aalst, *Multi-Phase Process Mining: Building Instance Graphs,* Lecture

Notes in Computer Science ed., vol. 3288, Berlin, 2004.

[15] W. v. d. Aalst, T. Weijters and L. Maruster, "Workflow Mining: Discovering Process Models From Event Logs," *IEEE Transactions on Knowledge and Data Engineering,* vol. 16, no. 9, pp. 1128-1142, 2004.

[16] A. de Medeiros, W. v. d. Aalst and J. Weijters, "Workflow Mining: Current Status and Future," *On The Move to Meaningful Internet Systems. In Proceedings of CoopIS/DOA/ODBASE,* vol. 2888, no. i, pp. 389-406, 2003.

[17] A. Weijters, W. v. d. Aalst and A. de Medeiros, "Process Mining with the HeuristicsMiner-algorithm," BETA Working Paper Series, WP 166, Eindhoven University of Technology, Eindhoven, 2006.

[18] A. Weijters and J. Ribeiro, "Flexible Heuristics Miner (FHM)," BETA Working Paper Series, WP 334, Eindhoven University of Technology, Eindhoven, 2010.

[19] C. Günther and W. v. d. Aalst, "Fuzzy Mining - Adaptive Process Simplification Based on Multi-Perspective Metrics," in *Proceedings of BPM'2007, pp. 328-343*, Eindhoven University of Technology, Eindhoven, 2007.

[20] H. Reijers, S. Limam and W. v. d. Aalst, "Product-Based Workflow Design," *Journal of Management Information Systems,* vol. 20(1), p. 229–262, 2003.

[21] W. v. d. Aalst, B. v. Dongen, J. Herbst, L. Maruster, G. Schimm and A. Weijters, "Workflow Mining: A Survey of Issues and Approaches," *Data and Knowledge Engineering,* vol. 47, pp. 237-267, 2003.

[22] B. v. Dongen and W. v. d. Aalst, "A Meta Model for Process Mining Data," *Proceedings of the CAiSE Workshops, part 2,* vol. 2, pp. 309-320, 2005.

[23] C. Günther, "XES Standard Definition," Fluxicon Process Laboratories, 2009.

[24] W. v. d. Aalst, A. ter Hofstede and M. Weske, "Business Process Management: A Survey," in *International Conference BPM*, Eindhoven, the Netherlands, 2003.

[25] IEEE Task Force on Process Mining, *Process Mining Manifesto,* www.win.tue.nl/ieeetfpm/, 2011.

[26] D. Power, "A Brief History of Decision Support Systems v4.0," 10 March 2007. [Online]. Available: http://DSSResources.com/history/dsshistory.html. [Accessed January 2012].

Appendix A Municipality processes

Evenementenvergunningen

Number of traces (dossiers): 399

Number of events (document groups): 11

Number of instances: 1083

ID	Document group
A	Aanvraag vergunning
B	Aanvullende gegevens
C	Besluit B&W
D	Brief afwijzing
E	Gegevens aanvullend onderzoek
F	Nota B&W
G	Ontvangstbevestiging
H	Overig stuk uitgaand
I	Rapport evaluatie
J	Vergunning
K	Verslag overleg
L	Dossier Afgesloten

	A	B	C	D	E	F	G	H	I	J	K	L
A	0	42	1	0	3	0	201	5	0	108	1	26
B	4	0	0	1	5	0	0	1	0	59	0	54
C	0	0	0	0	0	0	0	0	0	2	0	0
D	0	1	0	0	0	0	0	0	0	1	0	12
E	0	2	0	0	0	1	0	1	0	6	0	5
F	1	0	1	0	0	0	0	0	0	0	0	0
G	9	44	0	13	3	0	0	3	1	113	1	29
H	1	1	0	0	1	0	2	0	0	1	0	6
I	0	1	0	0	0	0	0	0	0	0	0	1
J	10	27	0	0	3	0	1	1	1	0	0	264
K	0	0	0	0	0	0	0	0	0	0	0	2
L	0	0	0	0	0	0	0	0	0	0	0	0

	A	B	C	D	E	F	G	H	I	J	K	L
A	0	0.808511	0.5	0	0.75	-0.5	0.909953	0.571429	0	0.823529	0.5	0.962963
B	-0.808511	0	0	0	0.375	0	-0.977778	0	-0.5	0.367816	0	0.981818
C	-0.5	0	0	0	0	-0.5	0	0	0	0.666667	0	0
D	0	0	0	0	0	0	-0.928571	0	0	0.5	0	0.923077
E	-0.75	-0.375	0	0	0	0.5	-0.75	0	0	0.3	0	0.833333
F	0.5	0	0.5	0	-0.5	0	0	0	0	0	0	0
G	-0.909953	0.977778	0	0.928571	0.75	0	0	0.166667	0.5	0.973913	0.5	0.966667
H	-0.571429	0	0	0	0	0	-0.166667	0	0	0	0	0.857143
I	0	0.5	0	0	0	0	-0.5	0	0	-0.5	0	0.5
J	-0.823529	-0.367816	-0.666667	-0.5	-0.3	0	-0.973913	0	0.5	0	0	0.996226
K	-0.5	0	0	0	0	0	-0.5	0	0	0	0	0.666667
L	-0.962963	-0.981818	0	-0.923077	-0.833333	0	-0.966667	-0.857143	-0.5	-0.996226	-0.666667	0

Waarderingssubsidies

Number of traces (dossiers): 332

Number of events (document groups): 31

Number of instances: 1398

ID	Document Group
A	Aanvraag subsidie
B	Aanvulling op aanvraag
C	Aanvulling verantwoording
D	Advies voor B-W
E	B-W adbvies over aanvragen
F	Beschikking definitief
G	Bezwaar tegen besluit afwijzing-verlening-bevoorschotting-vastst
H	Brief Herrinnering voor indienen aanvraag subsidie
I	Brief afwijzing subsidie
J	Brief definitieve beschikking
K	Brief ontvangstbevestiging aanvraag met verzoek om aanvulling
L	Brief ontvangstbevestiging aanvraag
M	Brief ontvangstbevestiging aanvulling aanvraag
N	Brief ontvangstbevestiging tussentijdse rapportage
O	Brief ontvangstbevestiging zienswijze met verzoek om aanvulling
P	Brief toekenning subsidie
Q	Brief uitbetaling voorschot
R	Brief vaststelling subsidie
S	Brief verlening subsidie
T	Brief wijziging besluit afwijzing-verlening-bevoorschotting-vast
U	Indiening zienswijze op voorgenomen besluit
V	Ontvangstbevestiging met verzoek om aanvulling
W	Ontvangstbevestiging
X	Overig stuk inkomend
Y	Overig stuk intern
Z	Overig stuk uitgaand
AA	Overleg gemeente-subsidient
AB	Toezending aanvraagformulier
AC	Tussentijdse rapportage
AD	Verantwoording
AE	Zienswijze
AF	Dossier Afgesloten

Table 17 Mapping process names

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF
A	0	7	0	0	2	2	0	0	3	11	8	126	14	0	0	0	0	52	0	2	0	1	0	5	18	3	1	14	0	0	0	36
B	1	0	0	0	0	1	0	0	0	5	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	4
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
D	0	0	0	0	0	1	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
E	3	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	1	3	0	0	0	0	0	1	1	2	1	1	0	0	0	3
F	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
G	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
H	19	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	3	0	0	0	1
I	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	3	0	0	0	0	0	0	2
J	28	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	17
K	41	0	0	0	0	0	0	1	1	0	0	0	5	0	0	0	0	14	0	1	0	0	0	2	1	1	0	0	0	0	0	0
L	10	0	0	0	1	0	0	0	1	0	9	0	1	0	0	0	0	123	1	0	0	0	0	10	8	3	0	1	0	0	0	3
M	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	1	2	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
O	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
R	13	0	0	0	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	20	6	10	0	1	0	0	0	191
S	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	2
U	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V	11	2	0	1	0	0	0	0	0	25	0	0	0	0	0	1	0	0	0	0	0	0	0	3	2	1	0	0	0	0	0	1
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
X	5	1	1	1	1	0	0	1	0	2	1	1	0	1	0	0	0	12	0	0	0	1	0	0	6	9	1	3	1	0	0	23
Y	0	0	0	1	2	0	0	6	0	1	5	7	0	0	0	0	0	16	3	1	0	0	0	2	0	5	0	0	0	1	0	23
Z	3	1	0	0	1	1	0	0	1	1	0	1	0	0	0	0	0	7	0	0	0	0	0	10	1	0	0	2	0	1	0	13
AA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	3	0	0	0	0	0	0	0
AB	121	0	0	0	1	0	0	17	0	0	10	32	0	0	0	0	0	0	0	0	1	0	0	3	10	2	1	0	0	0	0	3
AC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
AD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
AE	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AF	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18 Direct succession matrix for the 'Waarderingssubsidies' process

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
A	0	0.666667	0	0	-0.166667	0.25	0	-0.95	0.75	-0.425	-0.66	0.846715	0.611111	0	-0.5	0
B	-0.666667	0	0	0	0	0.5	0	0	0	0.375	0	0	0	0	0	0.5
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0.5	0	0	-0.5	0.666667	0	0	0	0	0	0
E	0.166667	0	0	0	0	0	0	0	0.25	0	0	-0.5	0	0	0	0
F	-0.25	-0.5	0	-0.5	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0.95	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0
I	-0.75	0	0	0.5	-0.25	0	0	0	0	0	-0.5	-0.5	0	0	0	0
J	0.425	-0.375	0	-0.666667	0	0	0	0	0	0	0	0	0	0	0	0
K	0.66	0	0	0	0	0	0	0	0.5	0	0	-0.9	0.833333	0	0	0
L	-0.846715	0	0	0	0.5	0	0	-0.5	0.5	0	0.9	0	0.5	0	0	0
M	-0.611111	0	0	0	0	0	0	0	0	0	-0.833333	-0.5	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
O	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P	0	-0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	-0.5	0	0	0	0	0	0	0	0	0	0	0
R	-0.590909	0	0	0	-0.4	0	0.5	0	0	0	-0.933333	-0.976	-0.823529	0	0	0
S	0	0	0	0	0.666667	0	0	0	-0.5	0	0	-0.5	0	0	0	0
T	-0.666667	0	0	0	0	0	0	0	0	0	-0.5	0	0	0	0	0
U	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0
V	0.769231	0.25	0	0.5	0	0	0	0	0	0.961538	0	0	0	0	0	0
W	0	0	0	0	0	0	0	0	0	-0.5	0	0	0	0	0	0
X	0	0.5	0.5	0.5	0	0	0	0	0	0.25	-0.25	-0.75	-0.5	0.5	0	-0.5
Y	-0.947368	-0.5	-0.5	0	0.25	0	0	0.625	-0.75	0.5	0.571429	-0.0625	-0.666667	0	0	0
Z	0	0	0	0	-0.25	0.5	0	0	0.5	0.5	-0.5	-0.4	0	-0.5	0	0
AA	-0.5	0	0	0	-0.5	0	0	0	0	0	0	0	0	0	0	0
AB	0.786765	0	0	0	0	0	0	0.666667	0	0	0.909091	0.911765	0	0	0	0
AC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AE	0.5	0	0	0	0	0	0	0	0	-0.5	0	0	0	0	0	0
AF	-0.972973	-0.8	0	0	-0.75	-0.8	-0.5	-0.5	-0.666667	-0.944444	0	-0.75	0	0	0	0

Table 19 Dependency measures for the 'Waarderingssubsidies' process (part 1)

Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	
0	0.590909	0	0.666667	0	-0.769231	0	0	0.947368	0	0.5	-0.786765	0	0	-0.5	0.972973	A
0	0	0	0	0	-0.25	0	-0.5	0.5	0	0	0	0	0	0	0.8	B
0	0	0	0	0	0	0	-0.5	0.5	0	0	0	0	0	0	0	C
0	0	0	0	0	-0.5	0	-0.5	0	0	0	0	0	0	0	0	D
0.5	0.4	-0.666667	0	0	0	0	0	-0.25	0.25	0.5	0	0	0	0	0.75	E
0	0	0	0	0	0	0	0	0	-0.5	0	0	0	0	0	0.8	F
0	-0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	G
0	0	0	0	0	0	0	0	-0.625	0	0	-0.666667	0	0	0	0.5	H
0	0	0.5	0	0	0	0	0	0.75	-0.5	0	0	0	0	0	0.666667	I
0	0	0	0	0	-0.961538	0.5	-0.25	-0.5	-0.5	0	0	0	0	0.5	0.944444	J
0	0.933333	0	0.5	-0.5	0	0	0.25	-0.571429	0.5	0	-0.909091	0	0	0	0	K
0	0.976	0.5	0	0	0	0	0.75	0.0625	0.4	0	-0.911765	0	0	0	0.75	L
0	0.823529	0	0	0	0	0	0.5	0.666667	0	0	0	0	0	0	0	M
0	0	0	0	0	0	0	-0.5	0	0.5	0	0	0	0	0	0	N
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	O
0	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	P
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	Q
0	0	-0.5	-0.5	0	0	0	0.242424	-0.434783	0.166667	0	0.5	0	0	0	0.994792	R
0	0.5	0	0	0	0	0	0	-0.75	0	0	0	0	0	0	0.666667	S
0	0.5	0	0	0	0	0	0.5	-0.5	0	0	0	0	0	0	0.666667	T
0	0	0	0	0	0	0	0	0	0	0	-0.5	0	0	0	0	U
0	0	0	0	0	0	0	0.4	0.666667	0.5	0	0	0	0	0	0.5	V
0	0	0	0	0	0	0	0	0	0.666667	0	0	0	0	0	0	W
0	-0.242424	0	-0.5	0	-0.4	0	0	0.444444	-0.05	0	0	0.5	0	0	0.958333	X
0	0.434783	0.75	0.5	0	-0.666667	0	-0.444444	0	0.571429	-0.75	-0.909091	0	0	0	0.958333	Y
0	-0.166667	0	0	0	-0.5	-0.666667	0.05	-0.571429	0	0	0	0	0.5	0	0.928571	Z
0	0	0	0	0	0	0	0	0.75	0	0	-0.5	0	0	0	0	AA
0	-0.5	0	0	0.5	0	0	0	0.909091	0	0.5	0	0	0	0	0.75	AB
0	0	0	0	0	0	0	-0.5	0	0	0	0	0	0	0	0.5	AC
0	0	0	0	0	0	0	0	0	-0.5	0	0	0	0	0	0.5	AD
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AE
-0.5	-0.994792	-0.666667	-0.666667	0	-0.5	0	-0.958333	-0.958333	-0.928571	0	-0.75	-0.5	-0.5	0	0	AF

Table 20 Dependency measures for the 'Waarderingssubsidies' process (part 2)

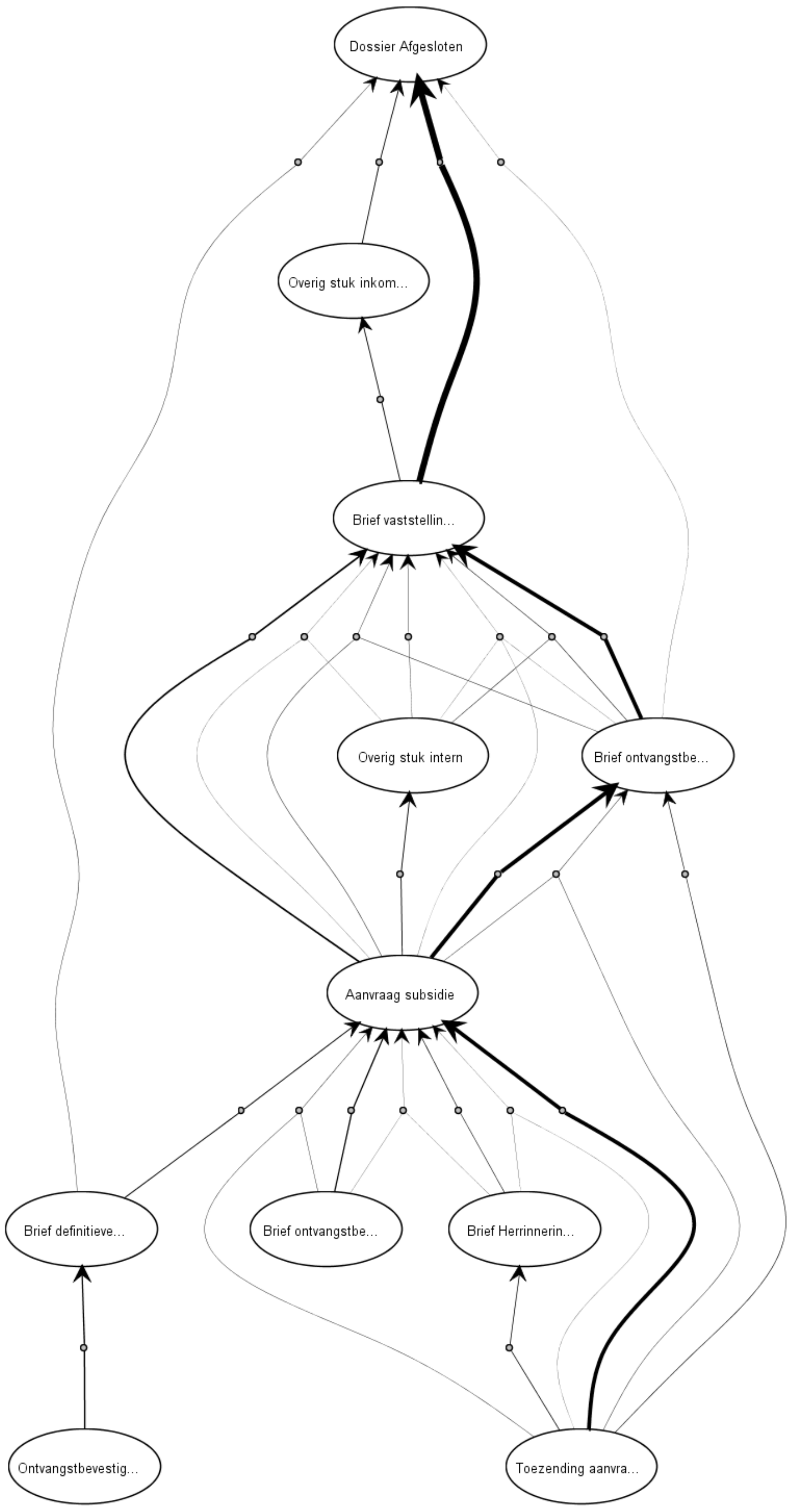


Figure 44 PDM for 'Waarderingssubsidies' process with a dependency threshold of 0.1

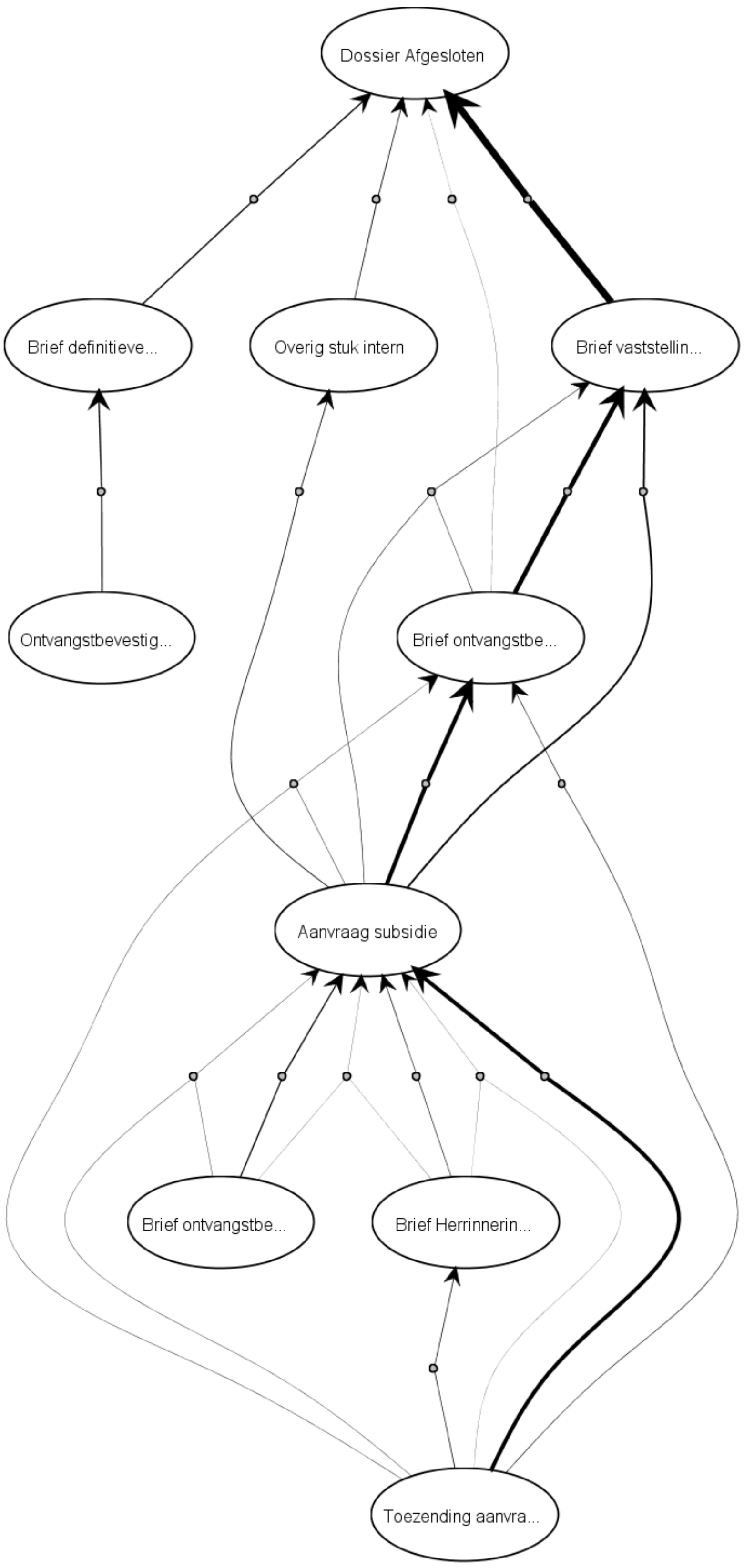


Figure 45 PDM for 'Waarderingssubsidies' process with a dependency threshold of 0.5

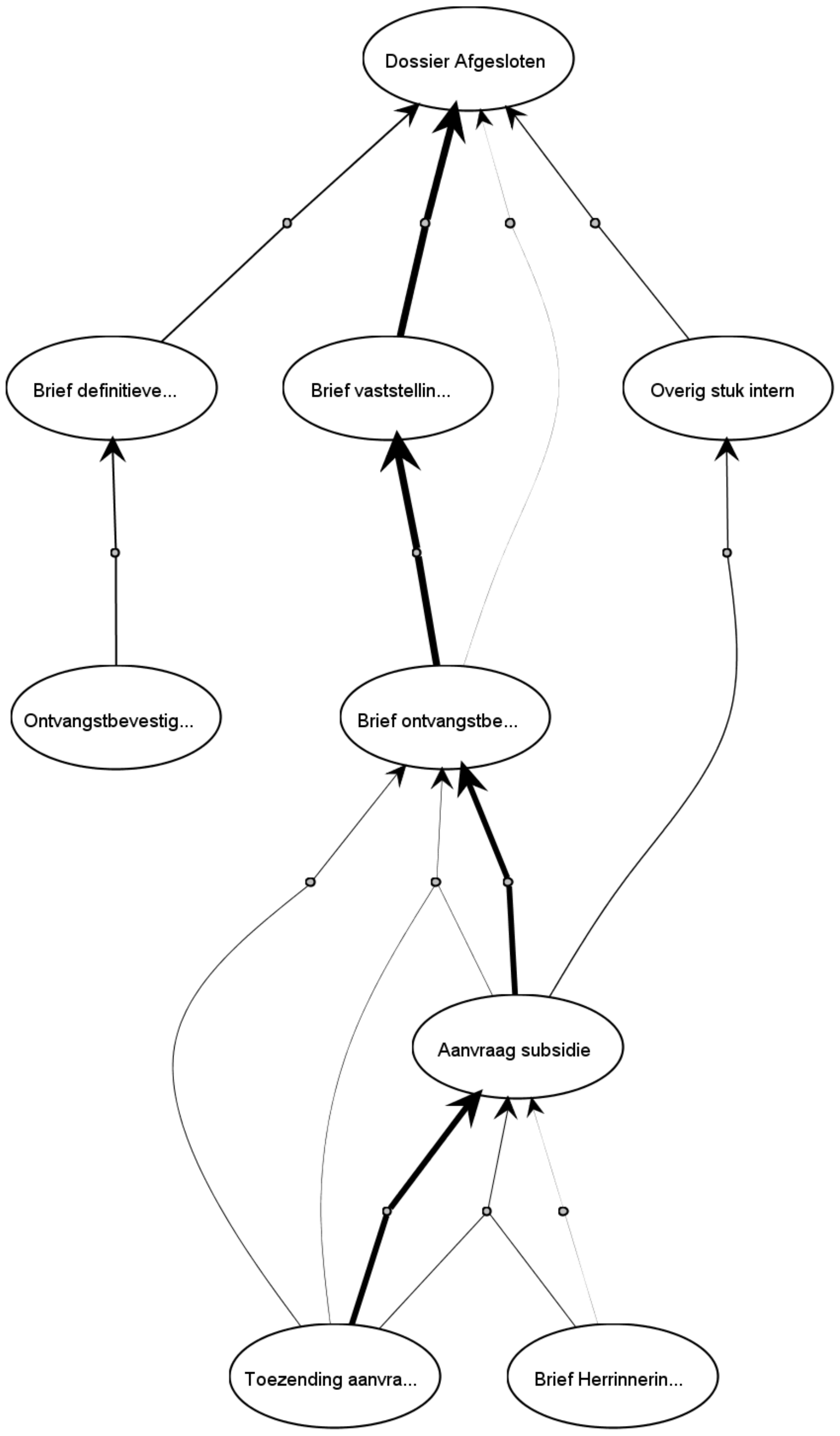


Figure 46 PDM for the 'Waarderingssubsidies' process with dependency threshold set to 0.75

Zienswijzen

Number of traces (dossiers): 361
Number of events (document groups): 4
Number of instances: 469

ID	Document Group
A	Ingekomen zienswijze
B	Ontvangstbevestiging
C	Overig stuk intern
D	Overig stuk uitgaand
E	Dossier Afgesloten

Table 21 Mapping of document groups

	A	B	C	D	E
A	0	53	5	17	285
B	6	0	4	12	41
C	0	0	0	7	3
D	2	2	0	0	32
E	0	0	0	0	0

Table 22 Direct succession relations

	A	B	C	D	E
A	0	0.783333	0.833333	0.75	0.996503
B	-0.783333	0	0.8	0.666667	0.97619
C	-0.833333	-0.8	0	0.875	0.75
D	-0.75	-0.666667	-0.875	0	0.969697
E	-0.996503	-0.97619	-0.75	-0.969697	0

Table 23 Dependency measures matrix

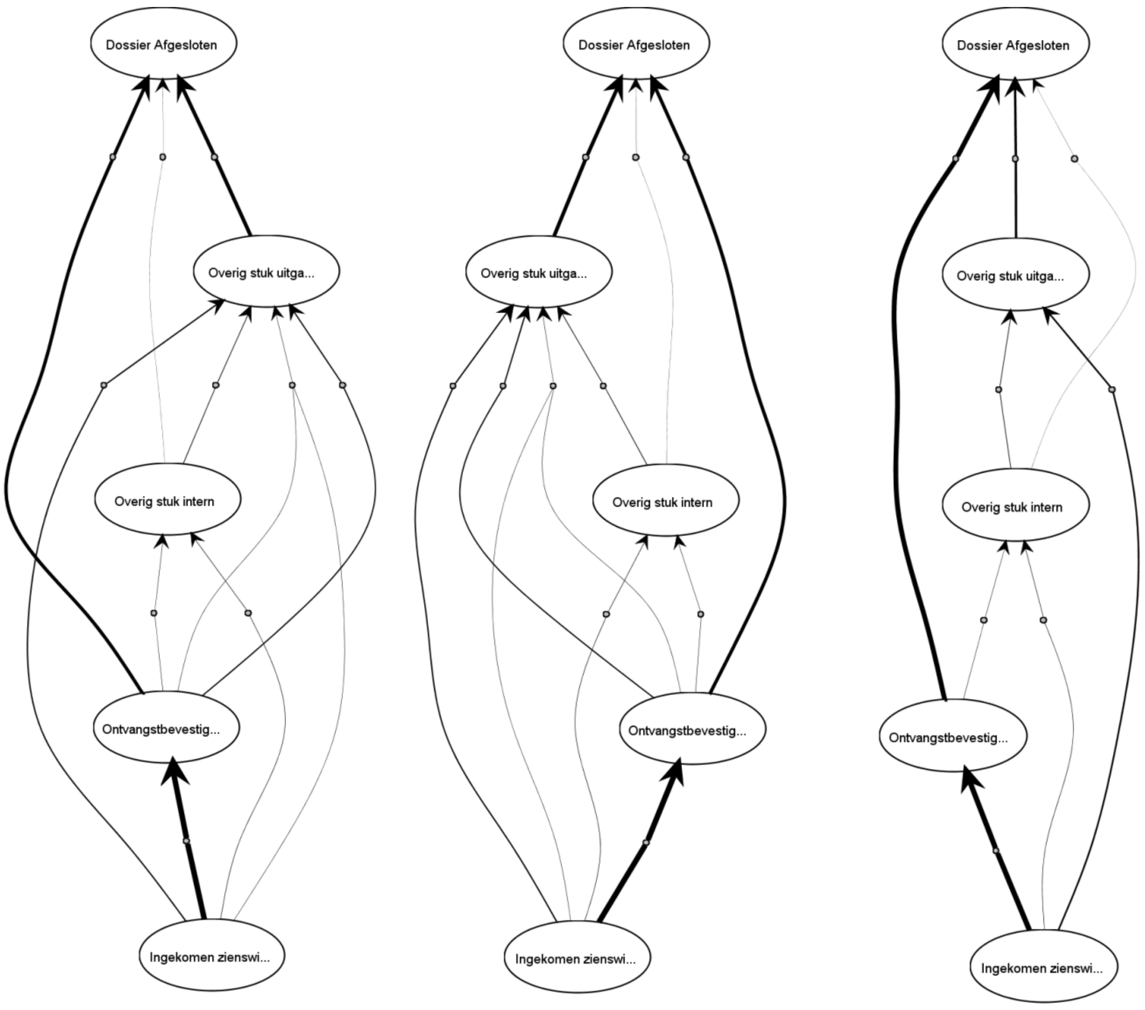


Figure 47 PDMs for 'Zienswijzen' process

Investeringssubsidies

Number of traces (dossiers): 35
Number of events (document groups): 39
Number of instances: 187

ID	Document Group
A	Aanvraag subsidie
B	Aanvraag vaststelling verleende subsidie
C	Aanvraag voorschot verleende subsidie
D	Aanvulling op aanvraag
E	Advies van commissie van bezwaarschriften
F	Advies voor B-W
G	B-W advies over aanvragen
H	B-W advies over advies van commissie van bezzwaarschriften
I	Beschikking toekenning
J	Bezwaar tegen besluit afwijzing-verlening-bevoorschotting-vas
K	Brief afwijzing subsidie
L	Brief beslissing op ingediend bezwaarschrift tegen besluit
M	Brief definitieve beschikking
N	Brief handhaving besluit afwijzing-verlening-bevoorschotting-vas
O	Brief herinnering indienen aanvraag tot vaststelling van verleen
P	Brief herinnering voor indienen aanvraag subsidie
Q	Brief ontvangsbevestiging aanvulling aanvraag
R	Brief ontvangstbevestiging aanvraag
S	Brief ontvangstbevestiging bezwaar met verzoek om aanvulling
T	Brief ontvangstbevestiging tussentijdse rapportage
U	Brief ontvangstbevestiging zienswijze
V	Brief vaststelling subsidie
W	Brief verlening subsidie
X	Brief verzoek aanvulling verantwoording
Y	Briefontvangstbevestiging aanvraag met verzoek om aanvulling
Z	Ingediende zienswijze op voorgenomen besluit
AA	Ontvangstbevestiging met verzoek om aanvulling
AB	Ontvangstbevestiging
AC	Overig stuk inkomend
AD	Overig stuk intern
AE	Overig stuk uitgaand
AF	Overleg gemeente-subsidient
AG	Raadsbesluit exploitatiesubsidie
AH	Raadsvoorstel exploitatiesubsidie

AI	Toezending aanvraagformulier
AJ	Tussentijdse rapportage
AK	Verantwoording
AL	Verweerschrift
AM	brief uitbetaling subsidie
AN	Dossier Afgesloten

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN
A	0	0	0	2	0	1	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	1	0	1	0	0	4	3	0	0	1	0	0	0	0	3	2
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	3	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
C	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0
D	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0	0	0	0	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
G	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	1
H	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
J	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
K	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
L	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
R	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	2	0	0	0	0	2	1	0	0	0	0	0	0	0	0	2
S	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Figure 48 Direct succession matrix for 'Investeringssubsidies' process (part 1)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
V	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	4
W	1	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0
X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Y	1	3	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
AA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
AB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
AC	0	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	10	1	0	0	0	1	0	0	0	4
AD	1	0	0	0	0	0	2	0	0	0	0	0	0	0	1	0	0	0	0	0	1	2	0	0	2	0	0	0	3	0	0	1	0	0	0	0	0	0	0	4
AE	2	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	2	2	0	0	0	0	0	0	0	0	0	11
AF	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	3	1	1	0	0	0	0	0	0	0	0	0
AG	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
AH	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
AI	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
AJ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
AK	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
AL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AM	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
AN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 49 Direct succession matrix for 'Investeringssubsidies' process (part 2)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
A	0	0	-0.5	0.666667	0	0.5	-0.5	0	0	0	0	0	0	0	0	0	0	0.75	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.666667	0	0
C	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0
D	-0.666667	0	0	0	0	0.5	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.5	0
F	-0.5	0	0	-0.5	0	0	0	0	0	0	0.25	0	0.5	0	0	0	0	0	0	0
G	0.5	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	-0.5	-0.5	0	0
H	0	0	0	0	0	0	-0.5	0	0	0	0	0.5	0	0	0	0	0	0	0	0
I	0	0	0	-0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	-0.25	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	-0.5	0	0	0	0	0	0	0	0	0	0	0	0.5
M	0	0	0	0	0	-0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.5	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0	-0.5	0	0
R	-0.75	-0.666667	-0.5	0	0	0	0.5	0	0	0	0	0	0	0.5	0	0	0.5	0	0	0
S	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	-0.5	0	0	0	0	0	0	0	0
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V	0	-0.4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0
W	0.5	0	0.666667	0	0	0	-0.5	0	0	0	0	0	0	0	0	-0.5	0.5	0	0	0
X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Y	0	0.75	0	0	0	0	0.5	0	0	0	0	0	0	0	-0.5	0	0	-0.666667	0	0
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AA	-0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AC	0	0.5	0.5	0.5	0	-0.5	0	0	0	0	0.5	0	0	0	0	0	0	0	0	-0.5
AD	-0.5	-0.5	0	0	0	0	0.25	0	0	0	0	0	0	0	0.5	-0.5	0	-0.666667	0	0
AE	-0.166667	0	0	-0.5	-0.5	0.5	0.5	0	0	0	0.5	0	0	0	0	0	0	0	0	0
AF	0.5	0	0	0	0	0	-0.5	0	0	0	0	0	0	0	0	0	0	0.5	0	0
AG	0	0	0	0	0	0	-0.5	0	0	0	0	0	0	0	0	0	0	0	0	0
AH	-0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AI	0	0	0	0	0	0	0	0	0	0.5	0	0	0	-0.5	0	0	0	0	0	0
AJ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AK	0	0	0	0	0	0	0	0	-0.5	0	0	0	0	0	0	0	0	0	0	0
AL	0	0	0	0	0	0	0	0	0	-0.5	0	0	0	0	0	0	0	0	0.5	0
AM	-0.75	0.5	-0.666667	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AN	-0.666667	0	0	0	0	0	-0.5	0	0	0	-0.75	0	-0.5	0	0	0	-0.5	-0.666667	0	0

Table 24 Dependency measures for the 'Investeringssubsidies' process (part 1)

	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN
A	0	0	-0.5	0	0	0	0.5	0	0	0.5	0.166667	-0.5	0	0.5	0	0	0	0	0.75	0.666667
B	0	0.4	0	0	-0.75	0	0	0	-0.5	0.5	0	0	0	0	0	0	0	0	-0.5	0
C	0	0	-0.666667	0	0	0	0	0	-0.5	0	0	0	0	0	0	0	0	0	0.666667	0
D	0	0	0	0	0	0	0	0	-0.5	0	0.5	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0	0	0.5	0	-0.5	0	0	0	0	0	0	0	0	0
G	0	0	0.5	0	-0.5	0	0	0	0	-0.25	-0.5	0.5	0.5	0	0	0	0	0	0	0.5
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0
J	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.5	0	0	0.5	0	0
K	0	0	0	0	0	0	0	0	-0.5	0	-0.5	0	0	0	0	0	0	0	0	0.75
L	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0
O	0	0	0	0	0.5	0	0	0	0	-0.5	0	0	0	0	0	0	0	0	0	0
P	0	0	0.5	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0
Q	0	0	-0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5
R	0	-0.5	0	0	0.666667	0	0	0	0	0.666667	0	-0.5	0	0	0	0	0	0	0	0.666667
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.5	0	0
T	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0
U	0	0	0	0	0	-0.5	0	0	0.5	-0.5	0	0	0	0	0	0	0	0	0	0.5
V	0	0	0	0	0	0	0	0	0.5	-0.666667	-0.5	0	0	0	0	0	0	0	-0.5	0.8
W	0	0	0	0	0.5	0	0	0	0.5	0	0	0	-0.5	0	0	0	0	0	0	0
X	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0
Y	0	0	-0.5	0	0	0.5	0	0	0	-0.666667	0.5	0	0	0	0	0	0	0	0	0.5
Z	0.5	0	0	0	-0.5	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0
AA	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0
AB	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0	-0.5	0	0	0
AC	-0.5	-0.5	-0.5	0	0	0	0	0	0	-0.4	0.615385	-0.4	0	0	0	0.5	0	0	0	0.8
AD	0.5	0.666667	0	-0.5	0.666667	0	-0.5	0	0.4	0	-0.666667	0	-0.5	0	0	0	0	0	-0.5	0.8
AE	0	0.5	0	0	-0.5	-0.5	0	-0.5	-0.615385	0.666667	0	-0.5	0	0	0	0	0	0	-0.5	0.916667
AF	0	0	0	0	0	0	0	0	0.4	0	0.5	0	0	0	-0.5	-0.5	0	0	-0.5	0
AG	0	0	0.5	0	0	0	0	0	0	0.5	0	0	0	-0.5	0	0	0	0	0	0
AH	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0
AI	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0
AJ	0	0	0	0	0	0	0	0	-0.5	0	0	0.5	0	0	0	0	0	0	0	0
AK	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0
AL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AM	0	0.5	0	0	0	0	0	0	0	0.5	0.5	0.5	0	0	0	0	0	0	0	0
AN	-0.5	-0.8	0	0	-0.5	0	0	0	-0.8	-0.8	-0.916667	0	0	0	0	0	0	0	0	0

Table 25 Dependency measures for the 'Investeringssubsidies' process (part 2)

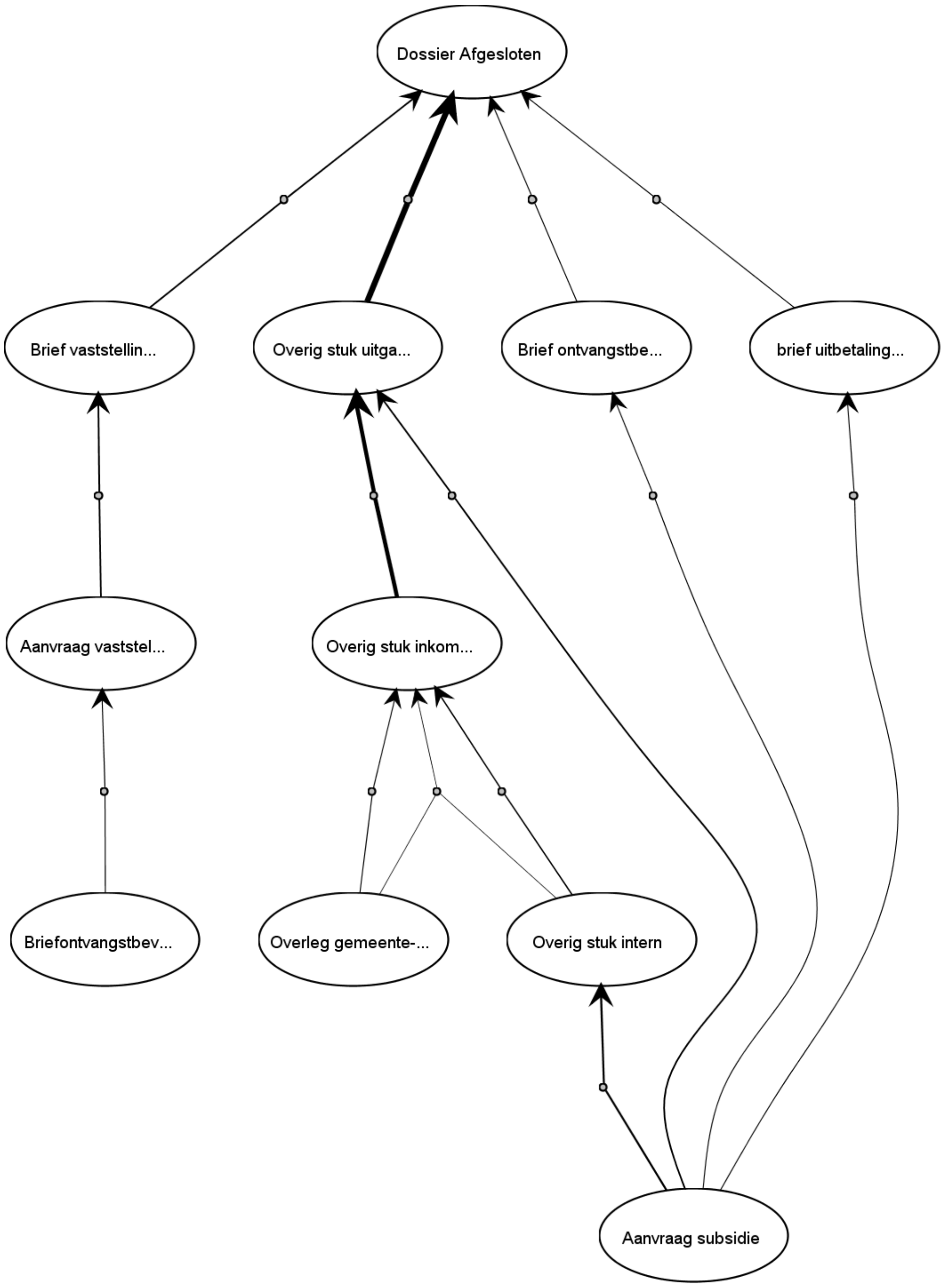


Figure 50 PDM for the 'Investeringssubsidies' process with dependency threshold set to 0.1

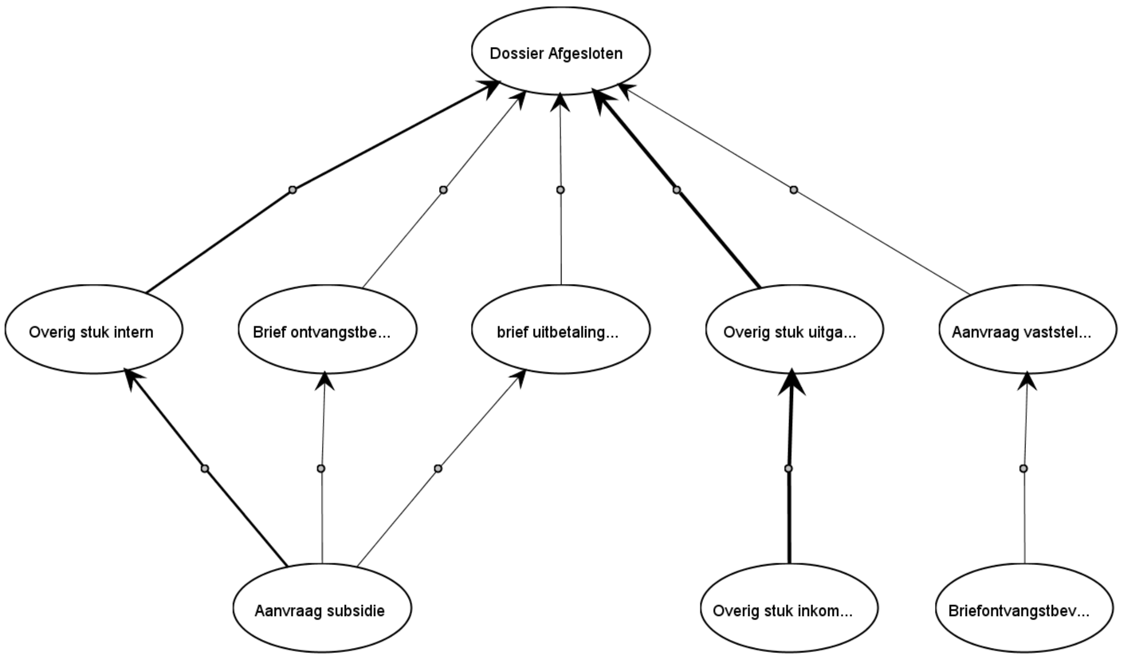


Figure 51 PDM for the ‘Investeringssubsidies’ process with dependency threshold set to 0.5

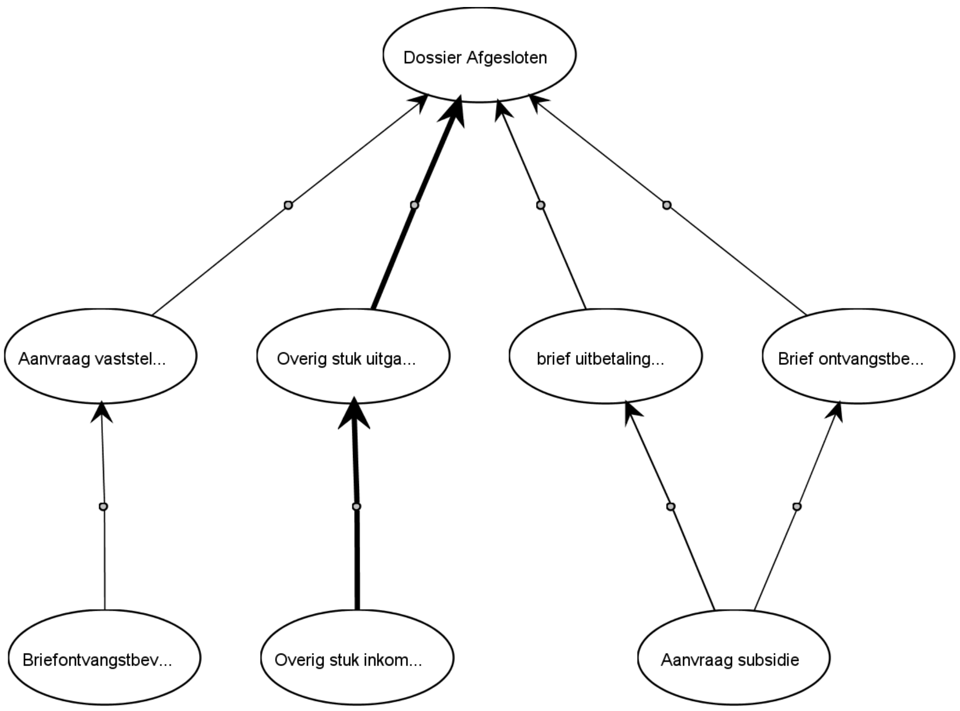


Figure 52 PDM for the ‘Investeringssubsidies’ process with dependency threshold set to 0.75

Gemeentelijk proces

Number of traces (dossiers): 1
Number of events (document groups): 4
Number of instances: 4

ID	Document Group
A	Advies definitief
B	Ingekomen post
C	Nota
D	Uitgaande post
E	Dossier Afgesloten

Table 26 Mapping of document groups

	A	B	C	D	E
A	0	0	1	0	0
B	0	0	0	0	1
C	0	0	0	1	0
D	0	1	0	0	0
E	0	0	0	0	0

Table 27 Direct succession relation for 'Gemeentelijk proces'

	A	B	C	D	E
A	0	0	0.5	0	0
B	0	0	0	-0.5	0.5
C	-0.5	0	0	0.5	0
D	0	0.5	-0.5	0	0
E	0	-0.5	0	0	0

Table 28 Dependency measures